우리식조작체계문고

Gambas

프로그람작성기초

```
PUBLIC SUB prn(str as string, s as float)
print str & "3      =" & s
END

STATIC PUBLIC SUB main()
Dim ss as float
ss = s(3,3,3)
prn("  ", ss)
prn("2  ", s(3,3,5))
prn(" ", s(3,4,5))
END
```

교육성교육정보쎈터
주체97(2008)년

차　　례

머 리 말

위대한 령도자 **김정일**동지께서는 다음과 같이 지적하시였다.

《프로그람을 개발하는데서 기본은 우리 식의 프로그람을 개발하는것입니다.》

(《**김정일**선집》 제15권, 196페지)

우리식의 조작체계를 개발할데 대한 경애하는 장군님의 말씀을 높이 받들고 우리의 과학자, 기술자들은 《붉은별》조작체계를 훌륭히 만들어 내놓았다. 그러나 우리앞에는 이러한 우리식 조작체계에서 리용할수 있는 더 많은 응용프로그람들을 개발하여야 할 중대한 과업이 나서고있다.

이 과업을 실현하자면 붉은별조작체계에서 프로그람개발도구들을 적극 리용하여야 한다.

붉은별조작체계의 프로그람개발도구에는 강력한 개발환경과 문법구조를 가진 Gambas가 있다.

Gambas는 Linux환경에서의 프로그람개발도구로서 Windows환경에서 사용되던 Microsoft Visual Basic와 아주 근사한것으로 하여 Linux환경으로 전환한 프로그람개발자들에게 매우 편리한 개발도구로 되고있다.

Gambas에서는 Visual Basic에서처럼 도형사용자대면부를 가진 프로그람을 개발할수 있을뿐아니라 Linux환경에서만 볼수 있는 조작탁응용프로그람도 작성할수 있다.

또한 자료기지관리, 인터네트와 봉사기들과의 련결, 자료압축과 같은 응용프로그람도 개발할수 있다.

뿐만아니라 Gambas응용프로그람은 통합개발환경을 통하여 서로 다른 Linux환경에서 리용할수 있는 실행파일로 번역될수 있다.

이와 같이 Gambas는 자기의 단순성을 유지하면서도 《전문》적인 개발도구로서의 모든 특징을 다 가지고있는것으로 하여 프로그람작성언어로서의 자기의 지위를 당당히 차지하고있다고 볼수 있다.

특히 Gambas는 Windows환경에서 Microsoft Visual Basic에 숙련된 개발자들에게는 Linux환경에서의 실질적인 프로그람개발도구를 제공하며 프로그람작성에 미숙한 사람들도 일정한 수준의 응용프로그람들을 개발할수 있게 하여준다.

독자들은 이 책을 학습하는 과정을 통하여 Gambas에 의한 프로그람작성수준을 한계단 더 높일수 있을것이다.

Gambas도 다른 개발도구들과 마찬가지로 계속 진화하고있다.

독자들은 이 책에서 나오는 실례들을 가지고 Gambas에 대한 충분한 련습을 진행함으로써 Linux환경에서 동작하는 응용프로그람들을 개발할수 있는 튼튼한 기초를 마련하여야 한다.

제 1 장. Gambas에 대한 리해

1. Gambas의 개발경위

Gambas는 프랑스의 어느 한 대학실습생이였던 베노이트 미니씨니(Benoit Minisini)에 의하여 초기에 개발되였다.

Gambas라는 이름 그 자체는 《Gambas almost means Basic》(Gambas는 Basic와 거의 비슷하다)에서 유래된것이며 베노이트는 Microsoft Visual Basic에 숙련된 개발자들을 위하여 이 프로그람을 개발하였다고 한다.

Visual Basic에서 쓸모있다고 생각되는 모든것 즉 Basic언어와 개발환경, 도형사용자대면부를 가진 프로그람을 빨리 작성할수 있는 기능 등을 하나하나 도입하여 Gambas를 개발하였다.

Gambas는 Linux조작체계를 탑재한 거의 모든 콤퓨터들에서 동작할수 있으며 이 책을 쓸 당시의 판번호는 1.9.23이다.

2. Gambas의 구성요소

Gambas로 작성된 모든 프로그람은 프로젝트파일들로 구성되여있다.

매개의 프로젝트는 클라스파일, 홈, 모듈과 자료파일들로 구성된다.

Gambas에서 프로젝트는 단일한 등록부에 보관된다.

Gambas 그 자체는 다음과 같은 프로그람들로 구성되여있다.

- 콤파일러
- 해석자
- 파일압축기
- 도형사용자대면부 부분품(component: 콤포넨트)
- 개발환경

먼저 Gambas**콤파일러**에 의하여 프로젝트의 클라스파일들을 콤파일한다.

프로젝트를 콤파일한 후에 수정된 클라스들이 존재하면 그 클라스들에 대하여서는 재콤파일해야 한다.

콤파일된것들가운데서 일부는 Gambas해석자에 의하여 실행될수 있다.

표준적으로 Gambas**해석자**는 조작탁에 기초한 프로그람이다.

우의 과정은 Java가 작업하는것과 매우 류사하다.

클라스들사이의 외부참조는 실행시 동적으로 해결된다.

Gambas**파일압축기**는 전체 프로젝트구조를 자체로 실행할수 있는 파일로 변환한다.

도형사용자대면부콤포넨트는 도형사용자대면부(GUI)를 창조하기 위해 리용한다.

현재 Gambas는 gb.qt콤포넨트를 리용하여 도형대면부를 작성하고있다.

Gambas가 다른 프로그람들과 구별되는 특징은 프로그람작성자들이 Gambas언어 능력을 확장할수 있는 확장요소(콤포넨트)들을 포함하고있다는것이다.

실례로 Gambas에는 gb.net, gb.db와 같은 확장콤포넨트들이 있다.

Gambas**개발환경**은 여러가지 응용프로그람을 편리하게 개발할수 있도록 보장하여 주는 프로그람묶음이다.

사실 Gambas의 개발환경은 Gambas언어의 능력을 시위하기 위하여 Gambas로 작성한것이라고 한다.

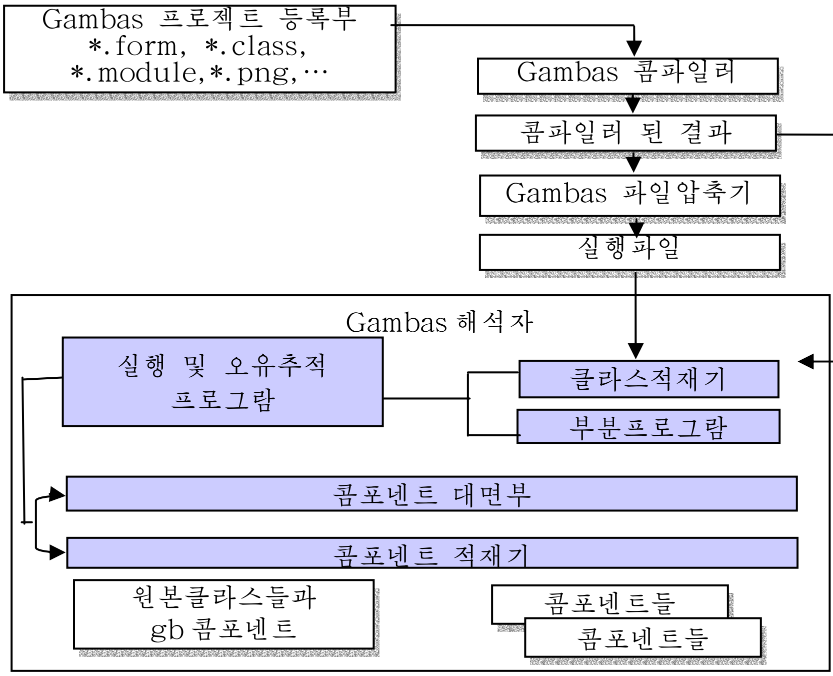


그림 1-1. Gambas의 전반적인 구성도

3. Gambas 프로그람작성환경

1) Gambas의 설치



그림 1-2. Gambas 프로그람패키지

Gambas프로그람패키지 gambas-1.9.23-13.i586.rpm을 두번 찰칵하여 실행시킨다.

그러면 프로그람추가 및 삭제창문이 현시된다. 《예》단추를 눌러 다음단계로 넘어가면 《의존성이 존재합니다.》라는 대화창이 현시된다. 《계속》단추를 찰칵하여 다음단계로 넘어간다. 이때 《잠간 기다리십시오, 설치중》

이라는 대화창이 나오면서 설치가 진행된다.

설치가 끝나면 설치가 정확히 완료되였다는 통보문이 현시된다.

2) Gambas의 기동

시작/응용프로그람/프로그람개발/Gambas를 찰칵하면 그림 1-3과 같은 화면이 펼쳐진다.

여기서는 다음과 같은 항목들가운데서 어느 한가지 항목을 선택할수 있다.

- 새로운 프로젝트를 창조(New project…)
- 이미 존재하는 프로젝트를 열기(Open project…)
- 최근에 리용한 프로젝트들의 열기(Recent projects)
- 실례프로그람들(Examples)
- Gambas의 탈퇴(Quit)

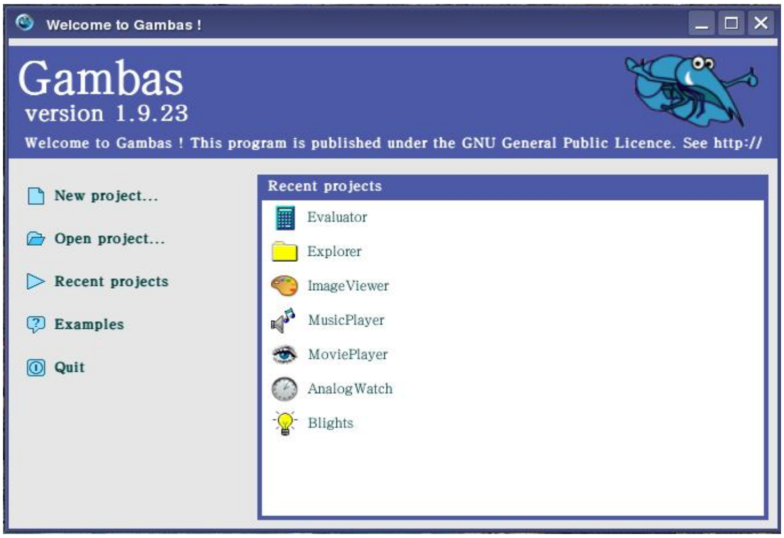


그림 1-3. Gambas열기창문

Gambas를 빨리 터득하기 위하여 New project…항목을 선택하여 첫 Gambas프로그람을 창조하여보자.

그러면 그림 1-4와 같은 대화창이 나타난다.



그림 1-4. Gambas프로젝트창조대화창

Next>>를 찰칵하여 다음단계로 넘어간다. 그러면 프로젝트의 형태를 요구하는 그림 1-5와 같은 대화창이 현시된다.

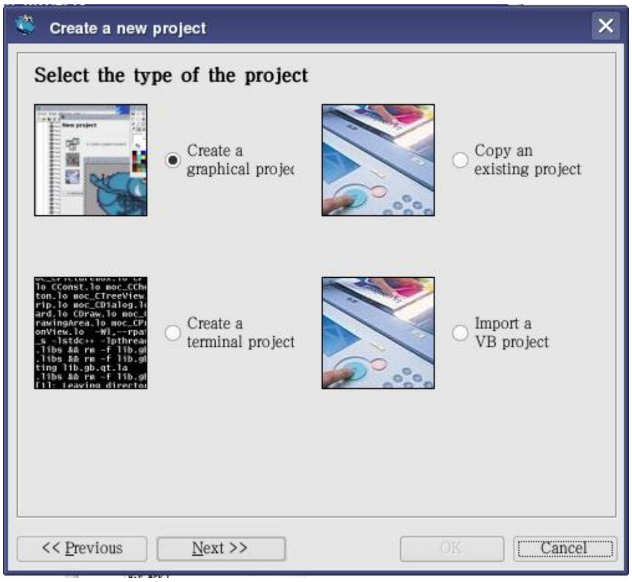


그림 1-5. 창조할 프로젝트의 형태를 선택하는 대화창

이 대화칸에는 다음과 같은 선택항목들이 있다.

● 도형사용자대면부를 가진 프로젝트를 창조하겠는가?(Creat a graphical project)

- 조작탁형프로젝트를 창조하겠는가?(Creat a terminal project)
- 복사하는 방법으로 프로젝트를 창조하겠는가?(Copy an existing project)
- VB프로젝트를 끌어들이는 방법으로 프로젝트를 창조하겠는가?(Import a VB project)

실례로 《Creat a graphical project》를 선택한 다음 Next>>를 찰칵하면 그림 1-6과 같은 대화창이 펼쳐진다.

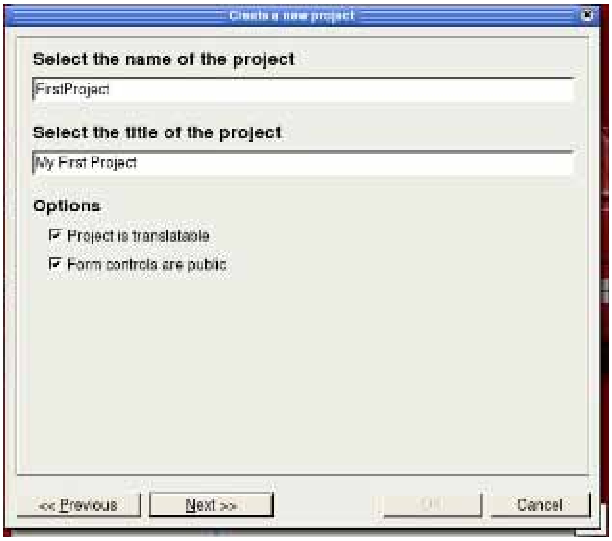


그림 1-6. 프로젝트창조대화창

이 대화창의 첫번째 본문칸에는 프로젝트의 이름을 입력한다.

Gambas에서 프로젝트는 여기서 지적한 이름을 가진 물리적파일로 등록부에 보관된다는것을 명심해야 한다.

대화창의 두번째 본문칸에는 프로그람창문의 제목띠에 표시되는 프로젝트의 제목을 입력할수 있다.

이 대화창에는 또한 2개의 선택항목 즉 project is translatable(프로젝트를 번역하겠는가?)과 Form controls are public(홈에서 리용된 콘트롤들을 public로 처리하겠는가?)가 있는데 이 항목에 대해서는 후에 구체적으로 설명한다.

적당한 이름을 입력한 다음 Next>>단추를 찰칵하여 다음단계로 넘어간다.

그러면 아래와 같은 프로젝트등록부선택대화창이 현시된다.(그림 1-7)

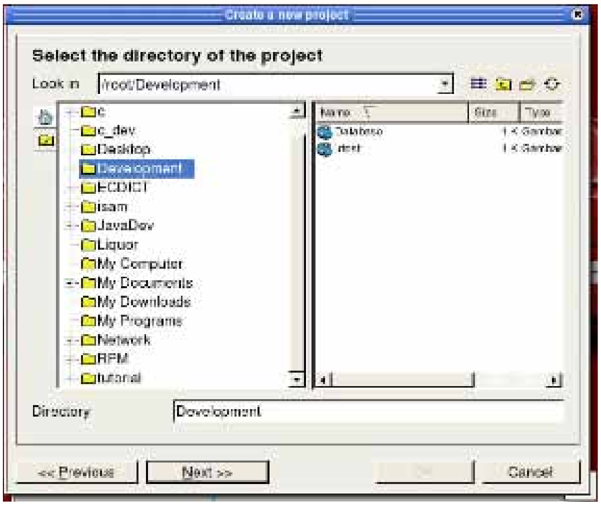


그림 1-7. 프로젝트 등록부선택대화창

여기서 프로젝트가 보관되게 될 등록부를 지적한 다음 Next>>단추를 찰칵하면 그림 1-8과 같은 대화창이 현시된다. 이 대화창은 새로 창조하는 프로젝트에 대한 자기의 요구를 최종적으로 확증하는 대화창으로서 이전단계에서 설정하였던 여러가지 항목들에 대한 변경을 진행할수 있는 마지막 기회로 된다.

OK단추를 찰칵하면 Gambas통합개발환경(IDE: Integrated Design Environment)이 펼쳐진다.

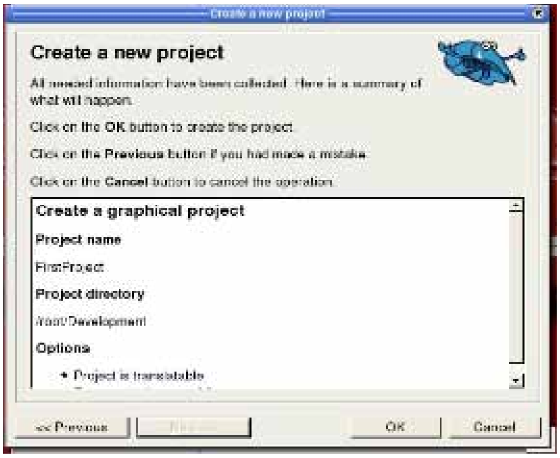


그림 1-8. 새 프로젝트 창조대화창

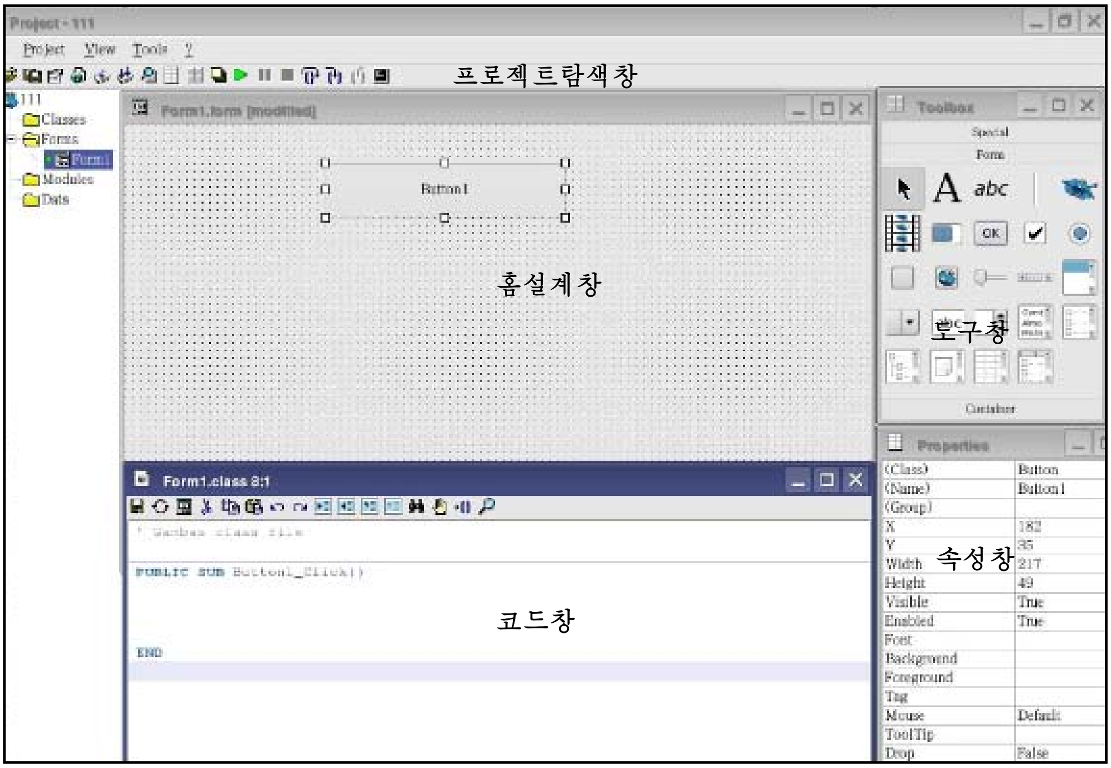


그림 1-9. Gambas통합개발환경



그림 1-10. File차림표

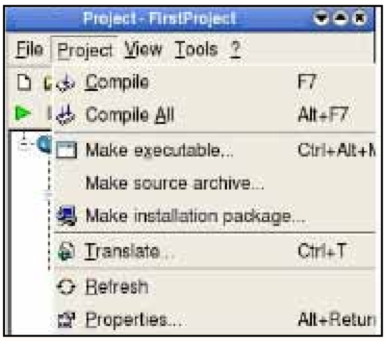


그림 1-11. project차림표

Gambas통합개발환경은 그림 1-9에서 보는바와 같이 여러 요소들로 구성되여있다.

3) Gambas의 IDE구성요소

① **프로젝트탐색창**은 Gambas기본창문이다.

프로젝트탐색창의 웃쪽에는 차림표와 도구띠들이 있다.

File차림표에서는 다음의것을 진행할수 있다.

- 새 프로젝트의 창조(New project)
- 프로젝트열기(Open project)
- 최근에 리용한 프로젝트열기(Open recent)
- 실례프로그람열기(Open example)
- 프로젝트보관(Save project)
- Gambas체계의 끝내기(Quit)

Project차림표에서는 다음의것을 진행할수 있다.

- 프로그람의 콤파일(Compile)

● 전체 프로젝트 콤파일(Compile All)
● 실행프로그람 만들기(Make executable)
● 원천프로그람압축(Make source archive)
● 설치프로그람묶음 만들기(Make installation package)
● 언어번역기능(Translate)
● 프로젝트속성설정(Properties)



그림 1-12. View차림표

View차림표에서는 다음과 같은 창문들의 보이기, 숨기기를 진행할수 있다

● 속성창(Properties)
● 도구창(Toolbox)
● 클라스의 계층구조(Hierarchy)
● 조작탁(Console)
● 아이콘편집기창(Icon editor)
● 모든 창문의 닫기(Close all windows)

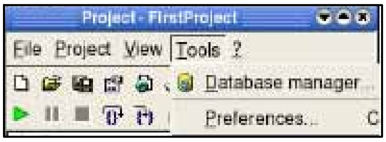


그림1-13. Tools차림표

Tools차림표에서는 다음의것을 진행할수 있다.

● 자료기지관리(Database manager)
● Gambas IDE환경설정(Preferences)

프로젝트탐색기의 나무구조로부터 다음과 같은 항목들이 현시되는것을 볼수 있다.

● Classes(클라스)

Classes : 사용자가 창조한 프로젝트의 클라스파일들을 렬거해준다.

클라스는 속성과 메쏘드, 사건조종콘트롤을 가진 객체들을 실행시에 창조할 때 리용되는 기초적인 모형들이다.

● Forms(홈)

Forms : 프로젝트를 위하여 창조한 여러가지 홈들을 렬거한다.

홈은 사용자가 실지로 호상 대화하게 될 창문이다.

● Modules(모듈)

Modules : 프로젝트에 작성된 모듈들을 현시해준다.

모듈은 클라스와 달리 프로그람의 임의의 곳에서 리용할수 있는 부분프로그람들과 함수들의 간단한 모임이다.

● Data(자료)

Data : 프로젝트내에서 리용되는 기타 자료파일들을 렬거해준다.

자료파일로서는 그라프파일들과 아이콘, 비트매프, 본문 혹은 HTML파일이나 다매체파일과 같은 여러 종류의 파일들을 포함한다.

프로젝트탐색기의 밑에는 Gambas가 현재 무엇을 하고있는가를 보여주는 상태띠가 있다.

② **홈설계창**

Gambas에서 프로젝트개발을 시작할 때에는 보통 기본홈으로부터 시작한다.

기본홈은 프로그람의 시작과 초기화가 일어나는 장소이며 사용자들이 응용프로그람을 실행시킬 때 보통 처음으로 맞다들리게 되는 대면부이다.

대면부설계는 홈설계창에서 진행한다.

홈에 콘트롤들을 배치하고 속성을 변화시키는 방법으로 사용자의 기호에 맞는 대면부를 설계한다.

③ **도구창**

도구창에는 대면부설계도구들인 콘트롤들이 아이콘형태로 있다.

④ **속성창**

여기서는 홈과 각종 콘트롤들의 이름, 너비, 높이, 배경색, 전경색을 비롯한 여러가지 속성들을 변화시킬수 있다.

매개 콘트롤들의 속성들을 설정하여줌으로써 콘트롤들의 출현과 움직임을 변화시킬수 있다.

⑤ **코드창**

사용자와 대면부와의 호상작용은 사건으로 귀착된다.

어떤 목적하는 사건에 대한 처리는 코드를 통하여 진행하는데 이 코드작성은 코드창에서 진행한다.

이 책에서 서술한 모든 실례프로그람들은 붉은별조작체계에서 동작하는 Gambas(version 1.9.23)에서 개발한것이다.

실례프로그람들은 Gambas를 설치할수 있는 모든 콤퓨터들에서 실행시킬수 있다.

제 2 장. Gambas기초개념

이 장에서는 Gambas언어의 기초적인 개념들을 학습한다. 즉 Gambas자료형과 상수, 변수의 정의방법과 상수, 변수들에 대한 값주기방법을 학습한다.

또한 기초적인 산수연산자들과 비교연산자들, 문자렬연산자들에 대하여서도 학습한다.

1. 변수

변수란 수학에서의 변수와는 달리 콤퓨터의 기억기에 일시적으로 자료가 보관되는 기억장소를 말한다.

프로그람을 작성할 때 필요에 따라 여러개의 변수를 사용할수 있다.

변수들은 클라스와 메쏘드 혹은 함수의 시작위치에서 정의하여야 한다.

변수는 틀이나 함수에서는 국부적으로 선언되지만 클라스에서는 대역적으로 선언될수 있다.

1) 국부변수

변수선언형식

[DIM] 변수이름 AS 자료형

실례

```
DIM iValue AS INTEGER
DIM stMyName AS STRING
DIM fMyMatrix AS FLOAT
```

이러한 변수들은 자기가 선언된 틀이나 함수에서만 리용할수 있다.

2) 대역변수

변수선언형식

[STATIC](PUBLIC|PRIVATE) 변수이름 AS [NEW] 자료형

Public를 지적하면 외부클라스들에서도 리용할수 있다는것을 의미한다.

Private를 지적하면 그 변수가 정의된 클라스의 내부에서만 리용할수 있다는것을 의미한다.

Static를 지적하면 그가 정의된 클라스의 모든 객체내에서 같은 이름을 가진 변수는 공유된다는것을 의미한다.

New를 지적하면 변수는 지적된 자료형에 따라 클라스의 새로운 실체로 초기화된다.

대역변수는 그가 선언된 클라스의 임의의 위치에서 리용될수 있다.

2. 자료형

프로그람작성자가 Gambas프로그람코드작성시에 리용할수 있는 기초적인 자료형들은 다음과 같다.

Boolean, Byte, Short, Integer, Float, Date, String, Variant, Object

1) Boolean형(론리형)

일반적으로 변수가 yes/no, TRUE/FALSE, 1/0값만을 가질 때 이 자료형을 리용한다. Boolean변수선언이 진행되면 변수가 기억기의 1byte만을 차지하며 표준값은 false이다.

[실례 1]

```
STATIC  PRIVATE  bGrid  AS  Boolean
```

2) Byte형(바이트형)

Byte형은 선언될 때 기억기에서 1byte를 차지하며 0~225까지의 값을 포함한다.
Byte형자료가 선언될 때 표준값은 0이다.
만일 255가 넘는 값을 리용할 때에는 이 자료형이 적합치 않다.
그러한 조건에서는 Short형이나 Integer형을 리용하는것이 더 좋다.

3) Short형(짧은옹근수형)

Short형값들은 -32 768 ~ +32 767범위의 값을 가진다.
Short자료형은 기억기에서 2byte를 차지하며 선언될 때 표준값은 0이다.

4) Integer형(옹근수형)

Integer자료형은 4byte를 차지한다.
Integer형에서 리용할수 있는 수값범위는 -2 147 483 648 ~ +2 147 483 647이다.
이것은 VB의 long형과 류사하다.

[실례 2]

```
STATIC  PUBLIC  sSomeShort  AS  Short
STATIC  PUBLIC  iSomeInteger  AS  Integer
```

Integer형자료가 적합하지 않을 때에는 Float형을 쓸수 있다.

5) Float형(류동소수형)

이 자료형은 프로그람에서 류동소수들을 쓸수 있게 한다.
Float형은 C나 VB에서 쓰던 Double형과 같다.

Float자료형의 범위는 부수로서는 -1.797 693 134 862 32E308 ~ -4.940 656 458 412 47E324, 정수로서는 4.940 656 458 412 47E324 ~ 1.797 693 134 862 32E308이다.

Float형변수들은 기억기에서 8byte를 차지하며 선언될 때에는 표준적으로 0값을 가진다.

[실례 3]

```
DIM fRadius AS Float
```

6) Date형(날자형)

Date형도 기억기에서 8byte를 차지한다.

날자값은 4byte옹근수로, 시간값은 4byte옹근수로 기억된다.

즉 [년, 달, 날자][시간, 분, 초]로 기억되며 보통 날자, 시간함수에서 많이 리용된다.

Date형은 선언될 때 표준으로 NULL값으로 초기화된다.

[실례 4]

```
DIM ddate AS Date
DIM dtime AS Date
```

String, Variant, Object자료형들은 Gambas에서 지원하는 비수값자료형들이다.

7) String형(문자렬형)

String형은 여러개 문자들의 련속으로서 문자자료들 혹은 빈문자렬을 포함한다.

여기서 문자자료란 글자나 수자 혹은 $%^&*와 같은 특수문자들을 의미한다.

String은 선언될 때 기억기에서 4byte를 차지하며 내적으로 문자렬의 길이를 표시한다.

[실례 5]

```
STATIC PUBLIC stSomeString AS String
```

8) Variant형(변화형)

Variant형은 받게 될 변수의 자료형을 알수 없을 때 리용한다.

실례로 만일 파일로부터 Integer형, String형, Float형수값자료를 읽는다고 할 때 프로그람의 파괴를 일으키지 않으면서 자료들을 변수에 넣자면 Variant형변수를 써야 한다.

이때 자료형을 결정하기 위하여 자료형검사함수들을 리용하여 자료형을 검증한 다음 변환함수를 리용하여 필요한 자료형으로 변환할수도 있다.

9) Object형(객체형)

Object형은 콘트롤이나 홈과 같은 객체들을 참조하는 특수한 자료형이다.

후에 객체지향프로그람작성에 대하여 론할 때 이 자료형의 리용에 대하여 구체적으로 설명하기로 한다.

다음의 표는 자료형에 대한 종합적인 자료들을 보여준다.

표 2-1. Gambas자료형

자료형	수범위	기억크기	표준값
Boolean	True or False	1byte	FALSE
Byte	0~255	1byte	0
Short	-32 768 ~ +32 767	2byte	0
Integer	-2 147 483 648 ~ +2 147 483 647	4byte	0
Float	C언어의 double형과 비슷함	8byte	0
Date	날자/시간	8byte	NULL
String	문자렬변수길이를 참조	4byte	NULL
Variant	임의의 자료형으로	12byte	NULL
Object	객체에 대한 간접적인 참조	4byte	NULL

지금까지의 변수선언실례들을 살펴보면 변수이름시작부분에 작은 글자《ar》,《s》,《i》, 《f》, 《d》, 《st》, 《v》, 《o》를 쓴것을 알수 있다.

이 문자들은 Gambas프로그람작성시 변수선언에서 공통적으로 리용되는 표시들로서 프로그람작성자들로 하여금 실천에서 변수에 대한 선언부분을 찾아보지 않고도 이름만 보고 변수의 형을 알수 있게 하여주는 수법이다.

일부 프로그람작성자들은 변수들의 이름을 특징짓는데 IntMyNumber, ByteSomething과 같이 더 많은 글자들을 리용하기도 한다.

3. 상수

프로그람실행과정에 변하지 않는 값을 상수라고 한다.

```
상수선언형식
    (PUBLIC|PRIVATE)  CONST  상수이름  AS  자료형  =
```

이 명령문으로는 클라스 대역상수들을 선언한다.

선언된 상수들은 클라스안의 임의의 장소에서 리용될수 있다.

만일 public를 지적하면 그 상수는 외부클라스에서도 리용할수 있다.

상수값들은 Boolean형, Integer형, Float형, String형이 될수 있다.

[실례 6]

```
PUBLIC  CONST  MAX  AS  Integer = 30
```

PRIVATE CONST MAGIC AS String = "Gambas form file"

표 2-2. Gambas 상수들

상 수	실 례
TRUE값	TRUE
FALSE값	FALSE
옹근수자료들	0, 562, 17, 32 769
16진부호있는 짧은 옹근수	&H100F3, &HF0FF, &FFFF
16진부호있는 옹근수	&H1ABF332E, &1CBF302E
16진부호없는 옹근수	&H80A0&, &HFCFF&
2진옹근수	&X1010111101, %101000011
류동소수	1.111 0, 5.341 9E+4
문자렬상수들	"Hello, Gambas World!"
빈문자렬	NULL

문자렬상수들은 다음과 같은 특수기호들도 포함한다.

표 2-3. 특수조종기호

특수조종기호	대응하는 ASCII값
\n	CHR$(13)
\r	CHR$(10)
\t	CHR$(9)
\”	인용괄호
\\	사선기호

문자렬상수들은 련속적으로 쓸수도 있다. 실례로 "선군정치" "만세"에 대하여 Gambas에서는 "선군정치만세"와 꼭 같이 인식한다.

4. 값주기명령문

다음과 같은 일반형식을 리용하여 변수에 임의의 값을 보관할수 있다.

● 일반값주기형식

```
일반값주기형식
        변수 = 식
            [실례]
                iMyVal = 1984
                stMyName = "Orwell"
                fMyNum = 123.45
```

● 객체속성에 대한 값주기

```
형식
        객체이름.속성 = 식
            [실례]
             hButton.X = 215
             hButton.Y = 60
             ProgressBar1.Value = 0.0
             Label2.Text = "some real"
```

같은 객체에 대한 속성값주기는 With명령문을 리용할수 있다.

```
형식
      WITH  객체
          .속성 = 식;

            ⋮
      END  WITH
          [실례]
           hButton.Width = 200
           hButton.Height = 40
           hButton.Enabled = TRUE
           hButton.Text = "Exit"

           이 명령문은 다음과 같이 쓸수 있다.

             WITH  hButton
               .Width = 200
               .Height = 40
               .Enabled = TRUE
               .Text = "Exit"
             END  WITH
```

이 명령문은 콘트롤들의 속성값을 줄 때 거의 공통적으로 사용된다.

5. 연산자와 식

지금까지 변수와 상수선언은 어떻게 하며 여기에 값주기는 어떻게 하는가에 대하여 보았다.

그러면 그것들사이에 수행될수 있는 연산자들을 보기로 하자.

먼저 비교연산자들을 보고 다음 산수연산자들과 문자렬연산자들을 보자.

1) 비교연산자

두 변수들의 비교는《x와 y는 같은가?》,《a는 b보다 작은가?》와 같은 질문에 대답을 줄것을 요구한다.

표 2-4. 비교연산자

연산자	의 미	실 례
=	같기	IF a = b THEN...
<>	같지 않기	IF a <> c THEN...
<	보다 작다	IF a < d THEN...
>	보다 크다	IF a > e THEN...
<=	같거나 작다	IF a <= f THEN...
>=	같거나 크다	IF a >= g THEN...

2) 산수연산자

가장 기초적인 산수연산자들로서 +(더하기), -(덜기), *(곱하기), /(나누기), \(옹근수나누기), mod(나머지), ^(제곱) 등이 있다.

값이나 변수에 덜기기호가 붙을수 있다. 례: -222, -aaa

[실전 실습]

첫 Gambas프로그람을 작성하여보자.

조작탁창문에서는 작성한 코드에 대한 실행결과를 즉시에 현시하여준다.

지금까지 학습한 자료형과 변수값주기에 기초하여 Gambas프로그람을 단계별로 작성해보자.

Gambas를 기동한 다음 File차림표에서 New project를 선택한다.

다음 New project창조조수에서 Create a Terminal Project(조작탁형프로젝트창조)를 선택하고 Next>>단추를 찰칵한다.

프로젝트이름편집칸에 TerminalTest를 입력한 다음 TerminalTest라는 등록부를 만들고 여기에 창조하려는 프로젝트를 보관한다.

다른 항목에 대해서는 상관없이 조수가 완성될 때까지 Next>>단추를 찰칵한다.

프로젝트탐색창의 나무구조에서 Classes항목을 찾아 마우스오른쪽단추를 찰칵한다.

이때 나오는 차림표에서 New/class...항목을 선택하면 New class대화창이 나온다.

체계설정이름인 class1을 그대로 두고 OK단추를 찰칵한다.

이때 코드창문이 현시되고 창문안에 다음과 같은 코드가 보인다.

```
' Gambas class file
  STATIC  PUBLIC  SUB  Main()
  END
```

코드의 첫행에 있는 웃반점은 설명문을 의미한다.

PUBLIC SUB Main()과 END사이에 다음과 같은 코드를 넣는다.

```
STATIC PUBLIC SUB Main()
  DIM N AS Integer
  DIM R AS Integer
   N = 3
   R = 6
  PRINT  "===> "  ;N; " | " ;-R; "과 "; -N ; " | " ;  R;
END
```

코드창에 코드를 입력한 다음 프로젝트탐색기의 도구띠에 있는 풀색단추(실행단추)를 찰칵하여 프로그람을 실행시킨다.

만일 다른 오유없이 실행되면 푸른색새우가 춤추는것을 볼수 있으며 조작탁창문에는 다음과 같은 결과가 나타난다.

===> 3 | -6 과 -3 | 6

[실례 7]

```
STATIC PUBLIC SUB Main()
 DIM N AS Integer
 DIM R AS Integer
  N = 8
  R = 5
 PRINT "===>" ; N-R;
END
```

결과:===> 3

[실례 8]

```
STATIC PUBLIC SUB Main()
```

```
  DIM N AS Integer
  DIM R AS Integer
   N = 8
   R = 5
  PRINT "===> " ; N * R;
END
```

결과:===> 40

[실례 9]

```
STATIC PUBLIC SUB Main()
  DIM N AS Integer
  DIM R AS Integer
   N = 9
   R = 3
  PRINT "===> " ; N / R;
END
```

결과:===> 3

만일 《/》다음에 오는 수의 값이 령이면 division by zero(령으로의 나누기)오유가 생긴다.

이때에는 stop단추를 찰칵하여 프로그람을 중지시킨다.

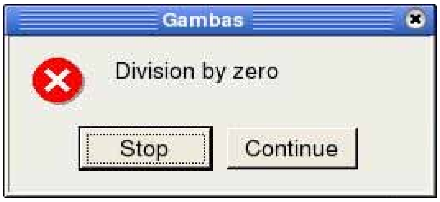


그림 2-1. 령으로의 나누기오유대화창

[실례 10]

```
STATIC PUBLIC SUB Main()
  DIM N AS Integer
  DIM R AS Integer
   N = 9
   R = 5
  PRINT "===>" ; N \ R;
END
```

결과:===> 1

나누기를 할 때 뒤로사선기호《\》를 리용하면 옹근수나누기가 진행된다.

[실례 11]

```
STATIC PUBLIC SUB Main()
 DIM N AS Integer
 DIM R AS Integer
  N = 9
  R = 5
 PRINT "===>상:"; N \ R; " 나머지: ";9 MOD 5;
END
```

결과:===>상: 1 나머지: 4

실례에서 보는바와 같이 mod연산자를 리용하여 두 수를 나눈 나머지를 구할수 있다.
여기서도 mod연산기호의 오른쪽에 놓이는 수가 령이면 오유가 발생한다.
제곱계산은 수^제곱형식을 리용하여 진행한다.

[실례 12]

```
STATIC PUBLIC SUB Main()
 DIM N AS Integer
 DIM R AS Integer
  N = 2
  R = 3
 PRINT "===>"; N ^ R;
END
```

결과:===> 8

DEC/INC는 변수값을 하나씩 감소/증가시킨다.
이때 변수는 반드시 수값형이여야 한다.

[실례 13]

```
STATIC PUBLIC SUB Main()
 DIM N AS Integer
 DIM R AS Integer
  N = 5
  R = 5
  DEC N
  INC R
```

```
  PRINT "===> " ; N; " | "; R;
END
```

결과:==> 4 | 6

Gambas에서는 C나 C++언어에서처럼 생략연산자를 쓸수 있다.
실례로 a+=2, b/=4

3) 론리연산자

Gambas에서는 또한 론리연산도 할수 있다.
론리연산자로서는 AND, OR, XOR, NOT가 있다.
AND를 리용하여 두수의 2진값에 대한 론리적을 계산한다.

[실례 14]

```
STATIC PUBLIC SUB Main()
 DIM N AS Integer
 DIM R AS Integer
  N = 0
  R = 0
  PRINT "=> "; N;"과 " ; R ;"의 AND 결과는: " ; N AND R
  R = 1
  PRINT "=> "; N;"과 " ; R ;"의 AND 결과는: " ; N AND R
  N = 1
  PRINT "=> "; N;"과 " ; R ;"의 AND 결과는: " ; N AND R
END
```

결과

```
=> 0 과 0 의 AND 결과는: 0
=> 0 과 1 의 AND 결과는: 0
=> 1 과 1 의 AND 결과는: 1
```

론리합연산은 OR를 리용하여 진행한다.

[실례 15]

```
STATIC PUBLIC SUB Main()
 DIM N AS Integer
 DIM R AS Integer
  N = 0
  R = 0
  PRINT "=> "; N;" 과 "; R ;" 의 OR 결과는: "; N OR R
```

```
    R = 1
    PRINT "=> "; N;" 과 "; R ;" 의 OR 결과는: "; N OR R
    N = 1
    PRINT "=> "; N;" 과 "; R ;" 의 OR 결과는: "; N OR R
END
```

결과

=> 0과 0의 OR 결과는: 0

=> 0과 1의 OR 결과는: 1

=> 1과 1의 OR 결과는: 1

XOR를 리용하여 두수의 배타적론리합을 계산한다.

[실례 16]

```
STATIC PUBLIC SUB Main()
  DIM N AS Integer
  DIM R AS Integer
    N = 0
    R = 0
    PRINT "=> "; N;" 과 "; R ;" 의 XOR 결과는: "; N OR R
    R = 1
    PRINT "=> "; N;" 과 "; R ;" 의 XOR 결과는: "; N OR R
    N = 1
    PRINT "=> "; N;" 과 "; R ;" 의 XOR 결과는: "; N OR R
END
```

결과

=> 0 과 0 의 XOR 결과는: 0

=> 0 과 1 의 XOR 결과는: 1

=> 1 과 1 의 XOR 결과는: 0

NOT연산자는 식의 론리적부정값을 계산한다.

형식
결과 =not 식

만일 식이 문자렬이거나 객체인 경우 식이 비여있지 않으면 TRUE를 돌려준다.

[실례 17]

```
STATIC PUBLIC SUB Main()
```

```
  PRINT NOT TRUE
  PRINT NOT FALSE
  PRINT NOT "Gambas"
  PRINT NOT " "
END
```

결과

```
False
True
False
True
```

Gambas에서 사용되는 연산자들의 연산순위 및 연산순서는 다음과 같다.

표2-5. 연산자들의 연산순위와 연산순서

연산자의 종류	연산자	순위	연산순서
산수연산자	^	1	왼쪽에서 오른쪽으로
	*, /, \, mod	2	왼쪽에서 오른쪽으로
	+, -	3	왼쪽에서 오른쪽으로
비교연산자	=,<,<=,>=,>	4	왼쪽에서 오른쪽으로
론리연산자	Not	5	왼쪽에서 오른쪽으로
	And	6	왼쪽에서 오른쪽으로
	Xor,or	7	왼쪽에서 오른쪽으로

4) 문자렬연산자

문자렬들을 비교하거나 련결할 때 문자렬연산자들을 리용한다.

Like는 문자렬의 론리적인 비교를 진행하는데 리용한다.

> 형식
>
> 문자렬 Like 패턴

만일 문자렬이 패턴과 꼭 같으면 TRUE값을 돌려준다.

패턴은 다음과 같은 패턴기호들을 포함할수 있다.

* 임의의 형태의 n개의 문자와 일치하면

? 어떤 하나의 문자와 일치하면

[abc] 괄호안에 들어있는 문자들중 어느 한 문자라도 일치하면

[x-y]　 x와 y사이에 있는 임의의 어느 한 문자와 일치하면
[^x-y]　 x와 y사이에 있지 않는 임의의 어느 한 문자와 일치하면

[실례 18]

```
STATIC PUBLIC SUB Main()
  PRINT "Rittinghouse" LIKE "R*"
END
```

결과:TRUE

문자렬에서 《\》다음에 나타나는 문자들을 단순한 문자렬로 생각해서는 안된다. 이것은 앞의 문자렬상수에서 본바와 같이 특수조종기호로 보아야 한다.

[실례 19]

```
STATIC PUBLIC SUB Main()
  PRINT "Samson" LIKE "S*"
  PRINT "Gambas" LIKE "?[Aa]*"
  PRINT "Leonard" LIKE "G[Aa]\\*"
  PRINT "Alfred" LIKE "G[^Aa]*"
END
```

결과

TRUE
TRUE
FALSE
FALSE

[주의]

패턴문자렬에 뒤사선기호 《\》를 기호로 리용하려면 뒤사선기호를 련속 2번 사용해야 한다. 그렇지 않으면 뒤사선기호 《\》를 '\n', '\t', '*'와 같은 특수조종기호로 해석할수 있다. 패턴문자렬에 《*》를 기호로 리용하려면 패턴문자렬을 "G[Aa][*]"와 같이 구성할수 있다.

다음의 문자렬연산자들은 문자렬들과 파일경로를 련결하거나 두개의 문자렬들을 비교할 때 리용할수 있다.

아래의것은 문자렬연산자표이다.

표 2-6.　　　　　　Gambas 문자렬연산자

문자렬연산자	실행결과
문자렬 & 문자렬	두 문자렬을 련결한다.

문자렬 &/ 문자렬	파일이름을 가진 두 문자렬을 련결한다. 필요하다면 두 문자렬들사이에 경로분리기호를 추가한다.
문자렬 LIKE 패턴	패턴에 따라 문자렬을 비교한다.
문자렬 = 문자렬	두 문자렬이 같은가를 결정한다.
문자렬 <> 문자렬	두 문자렬이 같지 않은가를 결정한다.
문자렬 < 문자렬	첫번째 문자렬이 두번째 문자렬보다 작은가를 결정한다.
문자렬 > 문자렬	첫번째 문자렬이 두번째 문자렬보다 큰가를 결정한다.
문자렬 <= 문자렬	첫번째 문자렬이 두번째 문자렬보다 같거나 작은가를 결정한다.
문자렬 >= 문자렬	첫번째 문자렬이 두번째 문자렬보다 같거나 큰가를 결정한다.

[실례 20]

```
STATIC PUBLIC SUB Main()
 DIM a AS String
 DIM b AS String
    a = "정보"
    b = "산업"
    PRINT "===> "; a & b;
    a = "Gambas"
    b = 2
    PRINT "===> "; a & b & " 가 개발되였습니다. " ;
END
```

결과

```
===> 정보산업
===> Gambas2 가 개발되였습니다.
```

Gambas는 사건지향언어이다.

프로그람작성자는 어떤 사건이 일어날 때 어떻게 해야 하는가를 결정할수 있는 능력을 가져야 하며 또한 Gambas의 예약어들을 리용하여 프로그람이 해야 할 일을 지적해주는 명령문들을 작성해나간다.

코드작성에서 먼저 맞다들리게 되는 예약어들을 간단히 소개한다.

① PRINT

이 지령은 식을 표준출력으로 출력한다.

```
형식
    PRINT 식 [(;|,) 식 ...] [(;|,)]
```

우에서 정의한 문법에서 사용된 괄호[]는 선택적인 인수라는것을 의미한다.

만일 마지막 식 다음에 반두점이나 반점이 없으면 마지막 식이 표시된 다음 새행으로의 표시가 자동적으로 진행된다.

만일 식들사이의 구분을 반두점대신 반점을 리용하면 식들을 구분하기 위하여 출구값들사이에 TAB(6글자)간격을 두고 표시된다.

[실례 21]

```
STATIC PUBLIC SUB Main()
  DIM b AS Integer
     B = 1
     PRINT "===> " & "b is: " & b
     PRINT "===> " , "b is: " , b
     PRINT "===> " ; "b is: " & b
END
```

결과

```
===> b is: 1
===>         b is:        1
===> b is: 1
```

② END

END는 함수나 수속의 끝을 나타낸다.

END를 사용하는데서는 VB와 3가지 차이점이 있다.

VB에서 END는 모든 홈들과 파일들을 닫고 프로그람을 끝내지만 Gambas에서의 END는 VB의 End Function이나 End Sub처럼 작용한다.

즉 END명령문은 함수나 부분프로그람의 끝을 표시한다.

③ QUIT

Gambas에서 VB의 END명령문과 류사한 기능을 하자면 QUIT명령문을 사용해야 한다.

그러면 프로그람을 즉시에 끝낼수 있다.

즉 모든 창문들이 닫기고 기억기에 있던 모든것들이 가능한껏 완전히 해방된다.

④ RETURN

Gambas에서는 수속이나 함수에서 탈퇴하려고 할 때 RETURN명령문을 사용할수 있다.

```
형식
    RETURN [ 식 ]
```

RETURN명령문을 수행하면 수속이나 함수에 <식>의 값을 돌려주고 자기 일을 끝낸다.

제 3 장. 프로그람흐름조종

1. 무조건이행 및 조건갈래명령문

1) GOTO명령문

```
형식
    Goto  표식
       명령문들
    표식:
```

표식이 있는 위치에로 무조건 이행하는 명령문이다.

즉 표식은 GOTO명령문이 가야 할 목적지이다.

표식을 정의할 때 두 점과 표식사이에는 공백이 없어야 한다.

2) IF명령문

```
형식
    IF   조건식   THEN
         명령문들1
          …
        [ELSE   IF   조건식
          명령문들
         …]
        [ELSE
          명령문들
```

IF명령문은 결심채택에 리용된다.

첫 조건식값이 TRUE이면 《명령문들1》을 수행하고 만일 FALSE이면 ELSE IF 다음의 조건식을 판단한다.

이때 조건식값이 FALSE이면 또 ELSE IF 다음의 조건식을 판단하는 식으로 조건식이 TRUE될 때까지 찾는다.

TRUE인 조건을 찾으면 대응하는 명령문들을 수행하고 ENDIF다음의 명령문을 수행한다.

TRUE인 조건을 찾지 못하면 ELSE 다음의 명령문들을 수행한다.

비교를 하고 그에 기초하여 결심을 채택하게 하는 가장 일반적인 명령문들중의 하나이다.

[실례 1]

```
STATIC PUBLIC SUB Main()
 DIM b AS Integer
  B = 1
  IF b = 1 THEN
     PRINT "===> " & "Gambas 의 판본은 " & b;
    ELSE IF b <> 1 THEN
     PRINT " 인쇄하지 말것";
    ELSE
     PRINT "오유발생"
  ENDIF
 END
```

결과:===> Gambas 의 판본은 1

3) SELECT / CASE 명령문

```
형식
    SELECT  비교식
      [CASE 값1 [, 값2 ...]
      [CASE 값1 [, 값2 ...]
      [CASE ELSE | DEFAULT ...]
```

Select명령문은 식을 계산하고 그것을 매 CASE에 지적된 값과 비교한 다음 같은 경우에는 그 CASE명령문안에 있는 코드를 수행한다. 만일 식값이 아무런 CASE값과도 일치하지 않으면 DEFAULT나 CASE ELSE다음의 명령문을 수행한다.

SELECT/CASE명령문은 여러개의 IF/ELSE명령문을 쓰지 않고도 많은 식값을 비교해야 하는 코드블로크를 작성할수 있게 한다.

[실례 2]

```
STATIC PUBLIC SUB Main()
 DIM w AS Integer
 w = 1
 START: ‘(START:는 아래의 GOTO문들에서 이행하게 될 표식)
 PRINT "w의 값은 : " & w
 SELECT CASE w
   CASE 1
     INC w
```

```
        GOTO START
      CASE 2
        INC w
        GOTO START
      CASE 3
        INC w
        GOTO START
      CASE ELSE
        PRINT "이 W값은 아무런 조종도 받지 않는다.: " & w
   END SELECT
   PRINT " w의 마지막값 : " & w
END
```

결과

w의 값은 : 1

w의 값은 : 2

w의 값은 : 3

w의 값은 : 4

이 w값은 아무런 조종도 받지 않는다: 4

w의 마지막값 : 4

실례에서는 변수 w의 값이 4까지 증가하기때문에 CASE에서 그 값을 비교하다가 마지막에는 CASE ELSE블로크를 수행하게 된다.

CASE ELSE대신에 CASE DEFAULT를 쓸수도 있다.

2. 순환명령문

1) FOR/NEXT 명령문

> 형식
>
> FOR 변수 = 시작값 TO 끝값 [STEP 걸음값]
>
> ...
>
> NEXT

FOR명령문은 프로그람에서 공통적으로 리용되는 순환명령문이다.

STEP값은 순환에서 증가 혹은 감소되여야 할 간격의 크기를 정해준다.

먼저 변수에 시작값을 넣고 순환부안의 명령문들을 수행하고 걸음값만큼 변수값을 증가(감소)시킨다.

다음 변수값을 끝값과 비교하여 끝값을 넘지(작지)않았으면 순환부분을 반복하고 매번 변수값을 걸음값만큼 증가(감소)시킨다.

순환은 끝값을 넘으(작으)면 끝난다.

Step가 생략된 경우 걸음값을 1로 하여 순환한다.

이때 변수는 수값형(Byte형, Short형, Integer형, Floating형)이여야 하며 또한 반드시 국부변수로 선언되여야 한다.

순환명령문에서 걸음값이 정수이면서 시작값이 끝값보다 큰 경우, 걸음값이 부수이면서 시작값이 끝값보다 작은 경우의 순환은 전혀 일어나지 않는다.

[**실례** 3]

```
STATIC PUBLIC SUB Main()
 DIM I AS Integer
 DIM J AS Integer
  J = 1
  FOR I = 1 TO 21 STEP 3
   PRINT "순환회수: " & J & ", 순환변수 I의 값: " & I
   INC J
  NEXT
 PRINT "J 값: " & J & " , I 값: " & I
END
```

결과

```
순환회수: 1, 순환변수 I의 값: 1
순환회수: 2, 순환변수 I의 값: 4
순환회수: 3, 순환변수 I의 값: 7
순환회수: 4, 순환변수 I의 값: 10
순환회수: 5, 순환변수 I의 값: 13
순환회수: 6, 순환변수 I의 값: 16
순환회수: 7, 순환변수 I의 값: 19
J 값: 8, I 값: 22
```

우의 실례에서 일단 I값이 끝값 21을 초과하면 순환은 중지되며 코드는 NEXT다음 명령문을 실행한다.

초기에 Gambas를 학습할 때에는 실례로 주어지는 코드들을 작성하고 PRINT명령문을 리용하여 변수값들을 변화시키면서 계속 련습하도록 하시오.

조작탁을 잘 리용하면 변수값과 코드토막을 쉽게 검증할 수 있다.

우의 실례에서 걸음값을 3으로부터 1로 변화시켜보시오.

다음은 순환명령문을 FOR I=21 to 1 STEP 1로 변화시키고 프로그람을 실행시켜보시오.

Gambas에는 또한 집합, 묶음, 유한개의 클라스들에 그 반복회수를 미리 알지 못해도 값들을 반복하여 넣을수 있는 FOR EACH 순환명령문이 있다.

FOR EACH명령문에 대해서는 집합을 배울 때 충분히 학습하기로 한다.

2) DO [WHILE] LOOP명령문

```
형식
    DO [WHILE 조건식]
    ...
    LOOP
```

DO [WHILE] LOOP명령문은 조건을 만족하거나 순환안에서 EXIT를 만날 때까지 순환을 반복한다.

순환되는 코드부분은 DO와 LOOP사이에 있는 명령문들이다.

만일 WHILE이 지적되지 않은 경우에는 무한순환이 계속 진행되거나 일부 조건이 탈퇴조건을 만날 때까지 순환이 계속된다.

또한 WHILE이 지적된 경우에는 일단 조건식의 검사결과가 FALSE이면 순환이 중지된다.

다시말하여 조건식값이 TRUE인 동안은 순환이 계속 반복된다.

만일 순환이 시작될 때 초기에 조건식값이 FALSE이면 순환은 절대로 일어나지 않는다.

[실례 4]

```
STATIC PUBLIC SUB Main()
 DIM a AS Integer
  A = 1
  DO WHILE a <= 5
    IF a = 1 THEN
       PRINT "순환 " & a & "번째."
     ELSE
       PRINT "순환 " & a & "번째."
    ENDIF
    INC a
  LOOP
  DEC a
  PRINT
```

```
    PRINT "안녕: 순환총회수" & a & "번."
END
```

결과

순환 1 번째.
순환 2 번째.
순환 3 번째.
순환 4 번째.
순환 5 번째.
안녕: 순환총회수: 5번.

3) WHILE WEND 순환명령문

```
형식
    WHILE [조건식]
    ...
    WEND
```

이 순환명령문은 조건식이 TRUE인 경우 순환이 반복된다.
만일 초기에 조건이 만족되지 않으면 순환이 절대로 일어나지 않는다.
즉 DO WHILE LOOP와 WHILE...WEND명령문은 꼭 같다.

[**실례** 5]

```
STATIC PUBLIC SUB Main()
 DIM a AS Integer
  a = 1
  WHILE a <= 5
   IF a = 1 THEN
     PRINT "WHILE...WEND 순환 " & a & " 번째."
    ELSE
     PRINT "WHILE...WEND 순환 " & a & " 번째."
   ENDIF
   INC a
  WEND
  DEC a
  PRINT
  PRINT "WHILE...WEND 순환총수: " & a & " 번."
END
```

결과

```
WHILE...WEND 순환 1 번째.
WHILE...WEND 순환 2 번째.
WHILE...WEND 순환 3 번째.
WHILE...WEND 순환 4 번째.
WHILE...WEND 순환 5 번째.
WHILE...WEND 순환총수: 5 번.
```

4) REPEAT UNTIL 순환

형식

```
REPEAT
...
UNTIL [조건식]
```

이 순환명령문은 조건식이 만족될 때까지(즉 조건식이 FALSE인 동안) REPEAT와 UNTIL사이의 명령문들을 반복한다.

REPEAT명령문은 UNTIL의 조건식값이 초기에 TRUE인 경우에도 무조건 한번은 순환을 수행한다.

[실례 6]

```
STATIC PUBLIC SUB Main()
 DIM a AS Integer
  A = 1
  REPEAT
   IF a = 1 THEN
      PRINT "REPEAT  UNTIL 순환 " & a & " 번째."
     ELSE
      PRINT "REPEAT  UNTIL 순환 " & a & " 번째."
   ENDIF
   INC a
  UNTIL a > 5
  DEC a
  PRINT
  PRINT "REPEAT UNTIL 순환총회수: " & a & " 번."
END
```

결과

```
REPEAT UNTIL 순환 1 번째.
REPEAT UNTIL 순환 2 번째.
REPEAT UNTIL 순환 3 번째.
REPEAT UNTIL 순환 4 번째.
REPEAT UNTIL 순환 5 번째.
REPEAT UNTIL 순환총회수: 5 번.
```

3. 묶음

Gambas에서는 두가지 종류의 묶음을 사용할수 있다.

첫번째 형태는 이미 Basic언어에서 전통적으로 리용되던 형태로서 정적인 묶음을 말한다.

> **형식**
> DIM 변수이름[한계1, 한계2, ...] AS 자료형

우와 같은 형식의 묶음은 객체도 아니며 동적인 묶음도 아니다.

즉 이러한 묶음들은 일단 선언되면 원소수를 더 늘이거나 없앨수 없으며 오직 선언된 묶음원소들에 대하여 값을 설정하거나 얻어낼수 있다.

[실례 7]

```
STATIC PUBLIC SUB Main()
 DIM I AS Integer
 DIM ii AS Integer
 DIM iii AS Integer
 DIM narMatrix[3, 3, 3] AS Integer
 FOR I = 0 TO 2
 FOR ii = 0 TO 2
 FOR iii = 0 TO 2
  PRINT i, ii, iii & " ==> ";
  narMatrix[i, ii, iii] = i*9 + ii*3 + iii
  PRINT narMatrix[i, ii, iii]
 NEXT
 NEXT
 NEXT
```

```
END
```

결과

```
0 0 0 ==> 0
0 0 1 ==> 1
...
2 2 1 ==> 25
2 2 2 ==> 26
```

이 실레에서 묶음은 27개의 원소들에 0~26까지 범위의 옹근수가 차있다.

다음 두번째 형태는 JAVA프로그람작성언어에서 사용되는 묶음과 같은 동적인 묶음이다.

형식
DIM 변수이름 AS NEW 자료형[]

즉 JAVA언어에서처럼 묶음을 Integer[], String[], Object[], Date[], Variant[]로 선언하면 묶음에 동적으로 객체(원소)를 더 추가하거나 없앨수 있다.

이 JAVA형묶음들은 다만 1차원만을 가질수 있다.

묶음은 선언되는 초기에 항상 빈것으로 초기화된다.

그러므로 묶음처리를 위한 몇개의 메쏘드들도 있다.

실례로 동적인 묶음변수에 객체(원소)를 추가하려면 add메쏘드를 리용하여 진행할수 있다.

배렬변수이름.add(입력값)

다음 실레에서는 프로그람수행과정에 3개의 원소를 가진 묶음이 창조된다.

[실례 8]

```
DIM i AS Integer
ss = NEW Single[]
FOR i = 0 TO 2
   ss.add(i*2)
   PRINT "ss[" & I & "]=" & ss[i]
NEXT
```

결과

```
ss[0]=0
ss[1]=2
ss[2]=4
```

4. 집 합

집합은 객체들의 묶음이다.

실례로 콘트롤들의 집합, 자료기지들의 집합 등 많은 집합들이 있다.

FOR EACH명령문을 리용하여 집합에 들어있는 원소들을 하나하나 처리할수 있다.

형식
FOR EACH 변수 IN 집합 ... NEXT

FOR EACH명령문은 집합에 있는 매 객체를 선택하는것과 동시에 그 개수만큼 순환을 진행한다.

선택순서는 예견할수 없다.

[실례 9]

```
STATIC PUBLIC SUB Main()
 DIM MyDict AS NEW Collection
 DIM strElement AS String
  MyDict["absolute"] = 3
  MyDict["basic"] = 1
  MyDict["carpet"] = 2
  FOR EACH strElement IN MyDict
    PRINT strElement & ", " ;
  NEXT
END
```

결 과: 3, 1, 2,

다음의 형식은 집합이 서로 구분되는 객체들을 포함하고있을 때 사용한다.

형식
FOR EACH 집합 ... NEXT

실례로 자료기지질문의 결과객체를 들수 있다.

[실례 10]

```
DIM res AS result
Res = DB.Exec( “Select * From Mytable” )
For each Res
   Print Res!code, Res!Name
Next
```

자료기지의 리용에 대하여서는 앞으로 구체적으로 보기로 한다.

5. 함수와 수속

프로그람을 작성할 때 동일한 코드블로크내용이 각이한 장소에서 여러번 반복되는 경우가 있게 된다. 이때 이 반복되는 부분을 함수나 틀로 따로 작성하여놓고 리용해야 할 장소에서 호출할수 있다.

따라서 프로그람작성에서 함수와 틀을 잘 리용하면 같은 코드를 여러번 작성할 필요가 없으며 프로그람의 설계, 코드작성, 오유수정을 간단히 진행할수 있다.

1) 함수

함수는 수학에서의 함수개념과 마찬가지로 일정한 입구량에 대하여 결과값(출구량)을 가진다. 함수에는 Gambas체계가 표준적으로 가지고있는 표준함수와 리용자가 정의하여 리용할수 있는 리용자정의함수가 있다.

프로그람을 작성할 때 필요한 표준함수는 리용만 하면 된다.

개별적인 표준함수에 대하여서는 다음장들에서 구체적으로 보기로 한다.

표준함수로 해결할수 없을 때에는 사용자정의함수를 만들어 리용한다.

```
함수선언형식
  PUBLIC FUNCTION 함수이름(파라메터1,파라메터2,…) as 자료형
    명령문들
    ...
    return 식
  END
```

여기서 매 파라메터들은 《변수 as 자료형》의 형식으로 자료형을 지적해주어야 한다.

```
함수호출형식
    함수이름(파라메터1, 파라메터2,…)
```

[실례 11]

```
PUBLIC FUNCTION s(a AS single,b AS single,c AS single) AS float
  Dim p as float
  If a+b>c and c+a>b and b+c>a then
   P=(a+b+c)/2
   S=sqr(p*(p-a)*(p-b)*(p-c))
  Endif
   RETURN s
END
STATIC PUBLIC SUB main()
  Dim ss as float
   ss= s(3,3,3)
   print “바른3각형의 면적”; ss
```

```
    ss=s(3,3,5)
    print "2등변3각형의 면적"; ss
    ss=(3,4,5)
    print "직3각형의 면적"; ss
END
```

여기서 sqr()는 두제곱뿌리를 계산하는 표준함수이지만 s()는 3각형의 면적을 계산하는 사용자정의함수이다.

2) 수속

수속은 함수와 비슷하지만 결과값을 돌려주지 않는다는 점에서 차이가 있다.

```
수속선언형식
    PUBLIC sub 수속이름(파라메터 1,파라메터 2,…)
      명령문들
      ...
```

매 파라메터들은 함수에서처럼 자료형을 지적하여주어야 한다.

```
수속호출형식
     수속이름(파라메터 1, 파라메터 2,…)
```

[실례 12]

```
PUBLIC FUNCTION s(a AS single, b AS single, c AS single) AS float
  Dim p as float
   If a+b>c and c+a>b and b+c>a then
    P=(a+b+c)/2
    S=sqr(p*(p-a)*(p-b)*(p-c))
   Endif
    RETURN s
END

PUBLIC SUB prn(str as string, s as float)
  print str & "3각형의 면적=" & s
END
```

```
STATIC PUBLIC SUB main()
  Dim ss as float
   ss = s(3,3,3)
   prn( "바른", ss)
   prn( "2등변", s(3,3,5))
   prn( "직", s(3,4,5))
END
```

이 실례에서는 실례 10에서 반복되는 부분을 prn수속을 리용하여 간단히 하였다.

이 프로그람은 간단한 프로그람이므로 수속을 리용한 우점이 크게 알리지 않지만 큰 프로그람인 경우 함수와 수속을 잘 리용하면 프로그람을 간단히 구조화할수 있으며 개발에서 편리하다.

제 4 장. 문자렬조종과 자료형변환

프로그람작성과정에 문자렬처리와 자료형을 변환시켜야 할 경우에 자주 부딪치게 된다.

따라서 프로그람을 작성할 때 문자렬조종과 자료형변환에 대하여 개발언어가 제공하는 표준함수들을 잘 리용하는것이 매우 중요하다.

주어진 함수들을 능숙하게 활용할수록 더 효과적인 프로그람을 작성할수 있다.

1. 문자렬조종함수

Gambas에는 많은 문자렬함수들이 있다.

이 장에서는 조작탁대면부를 리용하여 다음과 같은 표준함수들에 대하여 학습하게 된다.

① Len

형식
Len (문자렬)

돌려주는 값은 옹근수이다.

Len은 문자렬의 길이를 돌려준다.

[실례 1]

```
STATIC PUBLIC SUB Main()
 DIM iStringLength AS Integer
 DIM sTestString AS String
  sTestString = "12345678901234567890"
  sStringLength = Len(sTestString)
  PRINT "==> " & iStringLength & "은 첫 검사문자렬의 길이"
  sTestString = "12345"
  iStringLength = Len(sTestString)
  PRINT "==> " & iStringLength & " 은 두번째 검사문자렬의 길이"
  sTestString = "12345678901"
  iStringLength = Len(sTestString)
  PRINT "==>" & iStringLength & "은 세번째 검사문자렬의 길이"
END
```

결과
==> 20 은 첫 검사문자렬의 길이
==> 5 은 두번째 검사문자렬의 길이
==> 11 은 세번째 검사문자렬의 길이

[주의]

Len함수로 문자렬의 길이를 셀 때 C언어나 일부 다른 언어에서처럼 0위치에서가 아니라 첫위치에서부터 세기 시작한다.

다음은 문자렬들사이의 변환에 대하여 보자.

② Upper$/Ucase$/Ucase와 Lower$/Lcase$/Lcase

형식
Upper$ (문자렬) UCase$ (문자렬)

돌려주는 값은 문자렬이다.

Upper$는 문자렬을 대문자로 변환시킨다.

Ucase$와 Ucase도 Upper$와 꼭같이 리용할수 있다.

Lower$는 문자렬을 작은 글자로 변환시킨다.

Lcase$와 Lcase도 Lower$와 꼭같이 리용할수 있다.

[실례 2]

```
STATIC PUBLIC SUB Main()
  DIM sStringLength AS Integer
  DIM sTest AS String

    sTestString = "abcdefg"
    PRINT "==> " & sTest & " 는 시작문자렬이다."
    PRINT "==> " & UCase$(sTest) & "는 대문자로 된 문자렬이다."
    PRINT "==> " & LCase$(sTest) & "는 소문자로 된 문자렬이다."
    PRINT "==> " & Upper$(sTest) & "는 다시 대문자문자렬로 되였다."
    sTestString = "123abc456def 789ghiZZZ"
    PRINT "==> " & sTestString & ": 두번째검사문자렬."
    PRINT "==> " & UCase(sTestString) & ":대문자문자렬."
    PRINT "==> " & LCase(sTestString) & ":소문자문자렬."
    PRINT "==> " & Upper(sTestString) & ":대문자문자렬로 변환."
    PRINT "==> " & Lower$(sTestString) & ":소문자문자렬로 변환."
```

```
END
```

결과

==> abcdefg 는 시작문자렬이다.
==> ABCDEFG 는 대문자로 된 문자렬이다.
==> abcdefg 는 소문자글자로 된 문자렬이다.
==> ABCDEFG 는 다시 대문자문자렬로 되였다.
==> 123abc456def 789ghiZZZ: 두번째검사문자렬.
==> 123ABC456DEF 789GHIZZZ : 대문자문자렬.
==> 123abc456def 789ghizzz : 소문자문자렬.
==> 123ABC456DEF 789GHIZZZ : 대문자문자렬로 변환.
==> 123abc456def 789ghizzz : 소문자문자렬로 변환.

③ Trim$, LTrim$, RTrim$

형식 Trim$(문자렬)

돌려주는 값은 문자렬이다.
Trim$은 문자렬의 량끝에서 모든 공백을 없애버린다.
LTrim$은 문자렬의 왼쪽끝에서 모든 공백을 없애버린다.
RTrim$은 문자렬의 오른쪽끝에서 모든 공백을 없애버린다.
Trim$은 Ltrim$과 Rtrim$을 동시에 호출하는 셈이다.

[실례 3]

```
STATIC PUBLIC SUB Main()
 DIM sTestString AS String
 DIM sResult AS String
 DIM iLength AS Integer

  PRINT "          1         2"
  PRINT "12345678901234567890"
  sTestString = " <abcdef>    "
  iLength = Len(sTestString)

  PRINT sTestString & ": 검사문자렬."
  PRINT "검사문자렬의 길이: " & Str$(iLength)
```

```
    sResult = Trim$(sTestString)
    PRINT sResult & "는 검사문자렬을 Trim$로 호출한 결과이다."
END
```

결과

```
         1         2
12345678901234567890
 <abcdef>      : 검사문자렬.
검사문자렬의 길이: 14
<abcdef>는 검사문자렬을 Trim$로 호출한 결과이다.
```

④ Left$

> 형식
>
> Left$ (문자렬 [, 길이])

돌려주는 값은 문자렬이다.

Left$는 문자렬의 첫위치에서부터 지적된 길이만큼의 문자렬을 돌려준다.

만일 길이를 지적하지 않으면 문자렬의 첫 문자를 돌려주며 길이가 부수로 주어지면 길이만큼을 제외한 나머지문자렬을 돌려준다.

⑤ Mid$

> 형식
>
> Mid$ (문자렬, 시작 [, 길이])

돌려주는 값은 문자렬이다.

Mid$는 문자렬에서 지적한 시작위치로부터 지적한 길이만큼을 포함하는 부분문자렬을 돌려준다.

길이가 지적되지 않으면 시작위치로부터의 모든 문자렬을 돌려주며 길이가 부수이면 시작위치로부터 지적한 길이만큼의 마지막 문자렬을 제외한 나머지 문자렬을 돌려준다.

⑥ Right$

> 형식
>
> Right$ (문자렬 [, 길이])

돌려주는 값은 문자렬이다.

Right$는 문자렬의 마지막에서부터 지적한 길이만큼의 문자렬을 돌려준다.

길이가 지적되지 않으면 제일 마지막기호를 돌려주며 길이가 부수로 지적되면 전체 문자렬에서 앞으로부터 지적한 길이만큼의 문자렬을 제외한 나머지 문자렬을 돌려준다.

[실례 4]

```
STATIC PUBLIC SUB Main()
 DIM sTestString AS String
 DIM sResult AS String
 DIM iLength AS Integer

  PRINT "          1          2"
  PRINT "12345678901234567890"
  sTestString = "abcdefghi"
  iLength = Len(sTestString)
  PRINT sTestString & " : 검사문자렬."
  PRINT "검사문자렬의 길이는: " & Str$(iLength) & "이다."
  sResult = Left$(sTestString, 3)
  PRINT sResult & " 는 검사문자렬에 대하여 Left$를 호출한 결과"
  sResult = Mid$(sTestString, 4, 3)
  PRINT sResult & " 는 검사문자렬에 대하여 Mid$를 호출한 결과"
  sResult = Right(sTestString, 3)
  PRINT sResult & " 는 검사문자렬에 대하여 Right$를 호출한 결과"
END
```

결과

```
          1          2
12345678901234567890
abcdefghi : 검사문자렬.
검사문자렬의 길이는: 9 이다.
abc 는 검사문자렬에 대하여 Left$를 호출한 결과
def 는 검사문자렬에 대하여 Mid$를 호출한 결과
ghi 는 검사문자렬에 대하여 Right$를 호출한 결과
```

⑦ Space$

형식
Space$ (길이)

Space$는 주어진 길이만한 공백을 문자렬로 돌려준다.

[실례 5]

```
STATIC PUBLIC SUB Main()
  DIM sTestString AS String
  DIM sResult AS String

    PRINT "          1          2"
    PRINT "12345678901234567890123456"
    sTestString = "a"
    PRINT sTestString & Space$(24) & "z"
END
```

결과

```
          1          2
12345678901234567890123456
a                        z
```

⑧ Replace$

형식
Replace$ (문자렬 , 패턴 , 교체문자렬)

돌려주는 값은 문자렬이다.

Replace$는 문자렬에서 패턴과 일치하는 문자들을 교체문자렬로 교체하고 그 결과를 돌려준다. 문자렬이 비여있으면 공백문자렬을 돌려주며 패턴이 비여있으면 문자렬이 그대로 돌려진다.

[실례 6]

```
STATIC PUBLIC SUB Main()
  DIM sTestString AS String
  DIM sResult AS String

    sTestString = "abc123ghi"
    PRINT sTestString & " :검사문자렬."
    sResult = Replace$(sTestString,"123", "def")
    PRINT sResult & ":Replace함수를 호출한 결과이다."
    sTestString = "a b c d e f g h i"
    PRINT sTestString & " :검사문자렬."
```

```
    sResult = Replace$(sTestString, " ", "-")
    PRINT sResult & ":Replace함수를 호출한 결과이다."
    sTestString = "\ta\tb\tc\tdef"
    PRINT sTestString & " :검사문자렬."
    sResult = Replace$(sTestString,"\t", "")
    PRINT sResult & ": Replace함수를 호출한 결과이다."
END
```

결과

```
abc123ghi :검사문자렬
abcdefghi :Replace함수를 호출한 결과이다.
a b c d e f  g h i :검사문자렬.
a-b-c-d-e-f-g-h-i: Replace함수를 호출한 결과이다.
a     b    c    def :검사문자렬.
abcdef:Replace함수를 호출한 결과이다.
```

※ 문자렬에서 《\t》는 Tab를 의미한다.

⑨ String$

형식
String$ (반복회수, 패턴)

String$는 패턴을 지적한 수만큼 반복하여 만든 문자렬을 돌려준다.

[실례 7]

```
STATIC PUBLIC SUB Main()
  DIM sTestString AS String
  DIM sResult AS String
    PRINT "          1         2"
    PRINT "12345678901234567890123456"
    sTestString = "a"
    PRINT sTestString & String$(24,".") & "z"
END
```

결과

```
          1         2
12345678901234567890123456
a........................z
```

⑩ Subst$

> 형식
> Subst$ (패턴, 교체문자렬 [, 교체문자렬])

Subst$는 패턴에서 &1, &2를 대응하는 차례로 교체문자렬로 각기 교체하여 그 결과를 돌려준다.

이 함수는 문자렬들에서 일부 부분을 교체하여 련결해야 할 때 대단히 쓸모있다.

[실례 8]

```
STATIC PUBLIC SUB Main()
 DIM sTestString AS String
 DIM sResult AS String
  sTestString = "abcdef"
  sResult = "ghijkl"
  PRINT "검사문자렬: " & sTestString
  PRINT Subst$("교체하는 문자렬은: &1",sResult)
END
```

결과

검사문자렬: abcdef

교체하는 문자렬은: ghijkl

⑪ InStr

> 형식
> InStr(문자렬, 부분문자렬 [, 시작위치])

문자렬에서 왼쪽으로부터 부분문자렬을 탐색하여 첫위치를 옹근수값으로 돌려준다.

만일 시작위치를 지적하면 시작위치로부터 찾기 시작하며 부분문자렬을 찾지 못하면 령을 돌려준다.

[실례 9]

다음의 코드를 통하여 이 함수의 기능에 대하여 잘 알수 있다.

프로그람에서는 매 공백의 위치를 찾아서 그 위치를 표시하려고 한다.

```
' Gambas class file
STATIC PUBLIC SUB Main()
 DIM sTest AS String
 DIM sResult AS String
 DIM iLength AS Integer
```

```
    DIM iPosition AS Integer
    DIM iNextCharPos AS Integer
    DIM iCounter AS Integer

     sTest= "abc def ghi"
     iPosition = Instr(sTest," ")
     PRINT sTest & ":검사문자렬"
     PRINT " 첫 공백의 위치: " & iPosition
     iNextCharPos = iPosition + 1
     iPosition = Instr(sTest," ",iNextCharPos)
     PRINT "다음 공백의 위치: " & iPosition
   END
```

결과

abc def ghi :검사문자렬
첫 공백의 위치: 4
다음 공백의 위치: 8

⑫ RinStr

형식 RinStr (문자렬, 부분문자렬 [, 시작])

RinStr는 문자렬에서 오른쪽에서부터 부분문자렬을 탐색하고 그의 첫위치를 돌려준다.

만일 시작위치가 지적되면 탐색은 시작위치에서 멎으며 부분문자렬을 찾지 못하면 령값을 돌려준다.

[실례 10]

```
   STATIC PUBLIC SUB Main()
    DIM sTestString AS String
    DIM iPosition AS Integer
     sTestString = "abc def abc def abc"
     PRINT "          1          2"
     PRINT "12345678901234567890"
     PRINT sTestString
     iPosition = RInstr(sTestString,"abc")
     PRINT
     PRINT "문자렬 abc 의 오른쪽 첫 위치: " & iPosition
```

```
END
```

결과

```
                1             2
12345678901234567890
abc def   abc def abc

문자렬 abc 의 오른쪽 첫 위치: 17
```

⑬ Split

형식
Split(문자렬 [, 분리기호, 생략기호])

Split는 분리기호를 리용하여 문자렬을 분리하고 분리된 부분문자렬묶음을 돌려준다. 분리과정에 문자렬에서 생략기호들은 제거된다.

Split함수는 오직 3개의 파라메터만을 취할수 있으므로 여러개의 분리기호를 사용하려고 할 때에는 그 기호들을 련결하여 두번째파라메터로 지적해야 한다.

표준적으로는 반점기호가 분리기호이며 생략기호는 없다.

[실례 11]

```
STATIC PUBLIC SUB Main()
 DIM aWordArray AS String[]
 DIM sWord AS String
 ' 공백을 분리기호로 한다.
 ' '을 생략기호로 한다.
 aWordArray = Split("This is feature of 'Gambas!'"," ","'")
 FOR EACH sWord IN aWordArray
   PRINT sWord
 NEXT
END
```

결과

```
This
is
feature
of
Gambas!
```

2. 자료형변환함수

① Asc

형식 Asc(문자렬 [, 위치])

Asc는 문자렬에서 지적한 위치에 있는 문자의 ASCII코드값을 돌려준다. 위치가 지적되지 않으면 첫 문자의 ASCII코드값을 돌려준다.

② Chr$

형식 Chr$(코드)

Chr$는 ASCII코드값에 해당한 문자를 돌려준다.

[실례 12]

```
' Gambas class file
STATIC PUBLIC SUB Main()
 DIM iCode AS Integer
 DIM sTestString AS String
 DIM iLength AS Integer
 DIM counter AS Integer
  sTestString = "Gambas is great."
  iLength = Len(sTestString)
  PRINT "문자렬의 길이:" & Str$(iLength)
  FOR counter = 1 TO iLength
   iCode = Asc(sTestString, counter)
   IF iCode <> 32 THEN
    PRINT iCode & " 의 문자: " & Chr(9) & Chr$(iCode)
   ELSE
    PRINT iCode & "의 문자: " & Chr(9) & "<공백>"
   ENDIF
  NEXT
END
```

결과

문자렬의 길이: 16

71의 문자: G
97 의 문자: a
109 의 문자: m
98 의 문자: b
97 의 문자: a
115 의 문자: s
32 의 문자: <공백>
105 의 문자: i
115 의 문자: s
32 의 문자: <공백>
103 의 문자: g
114 의 문자: r
101 의 문자: e
97 의 문자: a
116 의 문자: t
46 의 문자: .

※ 우의 실례에서 Chr(9)는 TAB문자를 표현하는데 리용된다.

③ Bin$

형식 Bin$(수[, 자리수])

주어진 수를 2진수로 변환하여 돌려준다.
만일 자리수가 지적되면 그 자리수에 해당하여 령까지 첨가한 수형태를 보여준다.

[실례 13]

```
STATIC PUBLIC SUB Main()
 DIM sBinaryCode AS String
 DIM counter AS Integer
 FOR counter = 0 TO 15
  sBinaryCode = Bin$(counter,4)
  PRINT sBinaryCode
 NEXT
END
```

결과

0000

0001
0010
0011
0100
0101
0110
0111
1000
1001
1010
1011
1100
1101
1110
1111

④ CBool

형식
CBool (식)

CBool은 식에 대한 론리값을 돌려준다.
만일 다음과 같은 조건들을 만나면 false값을 돌려준다.

- 식이 론리부정값을 가질 때
- 식의 값이 령일 때
- 문자렬의 길이가 령일 때
- 객체가 비여있을 때

이밖의 경우에 함수는 true값을 돌려준다.

[실례 14]

```
STATIC PUBLIC SUB Main()
 PRINT CBool(0) ; " " ; CBool(1)
END
```

결과

```
FALSE TRUE
```

⑤ CByte

형식 CByte (식)

CByte는 식을 바이트형으로 변환한 값을 돌려준다.
식은 먼저 옹근수형으로 변환된다.
다음 이 옹근수값이 바이트값범위(-32 767 ~ 32 768)를 벗어나면 끌부분이 생략된다.

[실례 15]

```
STATIC PUBLIC SUB Main()
 PRINT CByte("17")
 PRINT CByte(32769)
 PRINT CByte(TRUE)
END
```

결과

```
17
1
255
```

⑥ CDate

형식 CDate (식)

CDate는 식을 날자/시간형으로 변환하여 돌려준다.

[실례 16]

```
STATIC PUBLIC SUB Main()
 DIM sDateString AS String
  sDateString = CDate(Now)
  PRINT sDateString; " :시작시간."
  WAIT 1.0
  PRINT CDate(Now); " :1S 지연되였습니다."
  PRINT CDate(sDateString); " :시작시간."
  WAIT 1.0
  PRINT Now; " 2S 지연되였습니다."
END
```

결과

```
01/21/2008 20:52:40 :시작시간.
01/21/2008 20:52:41 :1S 지연되였습니다.
01/21/2008 20:52:40 :시작시간.
01/21/2007 20:52:42 :2S 지연되였습니다.
```

⑦ CFloat

형식 CFloat (식)

식을 류동소수형태로 변환한다.

[실례 17]

```
STATIC PUBLIC SUB Main()
  DIM sFloatString AS String
  DIM fFloatNum AS Float

  sFloatString = "0.99"
  fFloatNum = 0.01

  PRINT fFloatNum + CFloat(sFloatString)
  PRINT CFloat("3.0E+3")
END
```

결과

```
1
3000
```

⑧ CInt/Cinteger

형식 CInt (식) CInteger (식)

Cint는 Cinteger 와 류사하다.
두 함수는 식을 모두 옹근수로 변환한다.

⑨ CShort

형식 CShort (식)

CShort는 식을 짧은 옹근수형으로 변환한다.

식은 먼저 옹근수형으로 변환된 다음 이 옹근수값이 짧은 옹근수범위를 벗어나면 뒤부분이 생략된다.

다음의 실례는 실례 15를 약간 수정한것이다.

[실례 18]

```
STATIC PUBLIC SUB Main()
 DIM sFloatString AS String
 DIM fFloatNum AS Float

 sFloatString = "0.99"
 fFloatNum = 120.901

 PRINT fFloatNum + CFloat(sFloatString)
 PRINT CInt(fFloatNum); " 은 옹근수형으로 변환되였습니다."
 PRINT CShort(fFloatNum); " 은 짧은 옹근수형으로 변환되였다."
END
```

결과

```
121.891
120 은 옹근수형으로 변환되였습니다.
120 은 짧은 옹근수형으로 변환되였다.
```

⑩ CStr/CString

형식 CStr (식) CString (식)

CStr/Cstring은 식을 문자렬로 변환한다.

[실례 19]

```
STATIC PUBLIC SUB Main()
 DIM sFloatString AS String
 DIM fFloatNum AS Float
 fFloatNum = 120.901
 sFloatString = CStr(fFloatNum)
 PRINT sFloatString; " 는 현재 문자렬이다."
END
```

결과:120.901 는 현재 문자렬이다 .

⑪ Hex$

> 형식
>
> Hex$ (수 [, 자리수])

Hex$는 수식을 16진수로 변환하여준다.

만일 자리수가 지적되면 그 자리수에 해당하여 령까지 첨가한 수형태를 보여준다.

[실례 20]

```
STATIC PUBLIC SUB Main()
  DIM counter AS Integer

  FOR counter = 1 TO 15
    PRINT Hex$(counter,2);" ";
  NEXT
END
```

결과

01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

⑫ Conv$

> 형식
>
> Conv$(문자렬, 문자코드 1, 문자코드 2)

Conv$는 문자렬을 첫 문자코드로부터 두번째 문자코드로 변환한다.

문자코드는 "ASCII", "ISO-8859-1", "UTF-8","EBCDIC-US"과 같이 문자렬로 표현할수 있다.

Gambas해석자는 내부에 《UTF-8》문자코드를 리용하고있다.

체계에서 리용되는 문자코드는 체계호출에 의하여 알아낼수 있다.

Mandrake9.2연산체계에서 리용되는 문자코드는 《ISO-8859-1》이며 RedHat Linux연산체계에서 리용되는것은 《UTF-8》문자코드이다.

앞으로 거의 모든 Linux 체계들은 아마도 《UTF-8》문자코드에 기초할것으로 보고있다.

[실례 21]

```
STATIC PUBLIC SUB Main()
  DIM sStr AS String
  DIM iInd AS Integer
```

```
  sStr = Conv$("Gambas", "ASCII", "EBCDIC-US")
  FOR iInd = 1 TO Len(sStr)
   PRINT Hex$(Asc(Mid$(sStr, iInd, 1)), 2); " ";
  NEXT
END
```

결과

C7 81 94 82 81 A2

⑬ Val

> 형식
>
> Val (문자렬)

Val함수는 문자렬의 내용에 따라 문자렬을 론리형, 수자 혹은 날자형으로 변환한다.

변환알고리듬은 다음과 같다.

만일 문자렬이 날자 혹은 시간으로 해석되면 날자/시간형식으로 변환하며 문자렬이 류동소수로 해석되면 류동소수로 변환한다. 또한 옹근수로 해석되면 옹근수로 변환한다. 문자렬이 TRUE 혹은 FALSE이면 론리형값으로 변환한다.

이외의 모든 경우에는 NULL값을 돌려준다.

⑭ Str$

> 형식
>
> Str$ (식)

Str$함수는 식을 인쇄할수 있는 문자렬식으로 변환한다.

즉 Val()과는 반대이다.

[**실례** 22]

```
STATIC PUBLIC SUB Main()
  DIM sDateTime AS String
  DIM dDate AS Date

  PRINT Now; " :현재 체계시간."

  sDateTime = Str$(Now)
  PRINT sDateTime; " :현재 체계시간을 문자렬로 변환."
  PRINT Val(sDateTime); " :문자렬을 VAL함수를 리용하여 변환."
```

```
    dDate = Val(sDateTime)
    PRINT dDate; " :VAL함수에 의하여 날자형으로 변환."
    PRINT Str(dDate); " :날자형을 문자렬형으로 변환."
END
```

결과

```
08/21/2005 21:42:45 :현재 체계시간.
08/21/2005 21:42:45 :현재 체계시간을 문자렬로 변환.
08/21/2005 21:42:45 :문자렬을 VAL함수를 리용하여 변환.
08/21/2005 21:42:45 :VAL함수에 의하여 날자형으로 변환.
08/21/2005 21:42:45 :날자형을 문자렬형으로 변환.
```

⑮ Format$

형식
Format$ (식 [, 형식화])

Format$함수는 식을 형식화를 리용하여 문자렬로 변환한다.

형식화파라메터는 옹근수상수로 지적한 값과 사용자정의형식(문자렬로 표현된것)을 결합하여 지적할수 있다.

사용자정의형식은 다음과 같은 형식화문자들을 리용하여 정의할수 있다.

표 4-1. 형 식 화 기 호

형식화문자	의 미
"+"	수자의 부호를 표시한다.
"-"	수자가 부수일 때만 부호를 표시한다.
"#"	필요하다면 수자만을 표시한다.
"0"	필요하다면 령까지 채운 자리수에 해당하여 표시한다.
"."	소수점을 찍는다.
"%"	수에 100을 곱하고 %기호를 표시한다.
"E"	류동소수의 지수부분을 표시한다. 그리고 지수부의 부호는 항상 표시한다.
"yy"	두자리수를 리용하여 년을 표시한다.
"yyyy"	네자리수를 리용하여 년을 표시한다.
"m"	달을 표시한다.
"mm"	두자리수를 리용하여 달을 표시한다.
"mmm"	3문자 생략문자렬형식으로 달을 표시한다.
"mmmm"	달의 이름을 이름그대로 표시한다.

"d"	날자를 표시한다.
"dd"	두자리수를 리용하여 날자를 표시한다.
"ddd"	생략된 형식으로 요일을 표시한다.
"dddd"	요일을 이름그대로 표시한다.
"/"	날자분리기호를 표시한다.
"h"	시를 표시한다.
"hh"	두자리를 리용하여 시를 표시한다.
"n"	분을 표시한다.
"nn"	두자리수로 분을 표시한다.
"s"	초를 표시한다.
"ss"	두자리수로 초를 표시한다.
":"	시간분리기호를 표시한다.

[실례 23]

```
STATIC PUBLIC SUB Main()
  PRINT "수 형식화의 실례: "
  PRINT Format$(Pi, "-#.###")
  PRINT Format$(Pi, "+0#.###0")
  PRINT Format$(Pi / 10, "###.# %")
  PRINT Format$(-11 ^ 11, "#.##E##")
  PRINT
  PRINT "날자, 시간형식화의 실례:"
  PRINT
  PRINT Format$(Now, "mm/dd/yyyy hh:nn:ss")
  PRINT Format$(Now, "m/d/yy h:n:s")
  PRINT Format$(Now, "ddd dd mmm yyyy")
  PRINT Format$(Now, "dddd dd mmmm yyyy")
END
```

결과

```
수 형식화의 실례:
 3.142
+03.1416
31.4 %
-2.85E+11
```

날자, 시간형식화의 실례:

```
08/21/2005 21:55:14
8/21/05 21:55:14
Sun 21 Aug 2005
Sunday 21 August 2005
```

3. 자료형검사함수

변수를 처리할 때마다 어떤 자료형인가를 아는것이 중요하다.
Gambas는 그러한 처리를 위해 일정한 함수묶음을 제공하고있다.

표 4-2. 자료형검사함수

함수형식	의미
IsBoolean (식)	론리형인가?
IsByte (식)	바이트형인가?
IsDate (식)	날자형인가?
IsFloat (식)	류동소수형인가?
IsInteger(식)	옹근수형인가?
IsNull (식)	비였는가?
IsNumber (식)	수자형인가?
IsObject (식)	객체형인가?
IsShort (식)	짧은옹근수형인가?
IsString (식)	문자렬형인가?

만일 식의 자료형이 요구하는 자료형이면 주어진 함수는 TRUE값을 돌려주고 그렇지 않으면 FALSE값을 돌려준다.

[실례 24]

```
' Gambas class file
STATIC PUBLIC SUB Main()
  DIM sInputLine AS String
  DIM value AS Variant
```

```
DO WHILE sInputLine <> "quit"
 PRINT "==> ";
 LINE INPUT sInputLine
 IF sInputLine = "" THEN
  sInputLine = "<CRLF>"
 ENDIF
 PRINT "입구자료: "; sInputLine
 value = Val(sInputLine)
 PRINT
 IF IsBoolean(value) THEN
    PRINT sInputLine; " :론리형."
   ELSE IF IsDate(value) THEN
    PRINT sInputLine; " :날자형."
   ELSE IF IsInteger(value) THEN
    PRINT sInputLine; " :옹근수형."
   IF value = 0 OR value = 1 THEN
     PRINT ":론리형도 된다."
   ENDIF
   IF value > 0 AND value < 255 THEN
     PRINT ":바이트형도 된다."
   ENDIF

   IF value > -32767 AND value < 32768 THEN
     PRINT ":짧은 옹근수형도 된다."
   ENDIF
    PRINT
   ELSE IF IsFloat(value) THEN
     PRINT sInputLine; " :류동소수형."
   ELSE IF IsString(value) THEN
     PRINT sInputLine; " :문자렬형."
   ELSE IF IsNull(value) THEN
     PRINT sInputLine; " : NULL."
   ELSE
  PRINT sInputLine; " :기타 자료형."
```

```
    ENDIF
  LOOP
  PRINT
  PRINT "이상 끝!"
END
```

결과

```
==> true
입구자료: true

true :론리형.
==> 214
입구자료: 214

214 :옹근수형.
:바이트형도 된다.
:짧은 옹근수형도 된다.

==> 08/23/05 12:23:55
입구자료: 08/23/05 12:23:55
08/23/05 12:23:55 :날자형.
==> John
입구자료: John

John :문자렬형.
==> -32756
입구자료: -32756

-32756 :옹근수형.
:짧은 옹근수형도 된다.

==> quit
입구자료: quit
quit :문자렬형.

이상 끝!
```

⑯ TypeOf

```
형식
      TypeOf ( 식 )
```

TypeOf는 식의 자료형을 옹근수값으로 돌려준다.
Gambas는 자료형에 대한 상수값들을 미리 정의하고있다.
그 상수들은 다음과 같다.

표 4-3. 정의된 자료형상수들

자료형상수	값
gb.Null	빈값
gb.Boolean	론리형
gb.Byte	바이트형
gb.Short	짧은 옹근수형
gb.Integer	옹근수형
gb.Float	류동소수형
gb.Date	날자/시간형
gb.String	문자렬형
gb.Variant	변화형
gb.Object	객체형

제 5 장. 수학연산

Gambas에서 수학연산은 표준함수들과 유도된 함수들을 리용하여 진행된다.

표준함수는 보통 프로그람작성언어의 콤파일러에 내장되여있으며 특정한 CPU의 특징을 살릴수 있는 우점을 가지고있다.

Gambas에서 콤파일러의 최량화와 코드발생은 표준함수들을 충분히 집약화하였으며 따라서 계산에서 놀라운 속도를 보장하고있다.

유도함수들은 한편 이미 알려진 수학공식들과 알고리듬을 표준함수들에 적용하여 만든것이다. 일부 유도함수들의 표와 그에 대응하는 공식들은 이 장의 마지막부분에서 볼수 있다.

1. 연산순서

식을 리용하여 연산을 진행할 때에는 특별히 정해진 연산순서가 있다.

연산순서는 다음과 같다.

- 괄호안에 있는 모든 식
- 지수연산은 왼쪽에서부터 오른쪽순서로
- 곱하기와 나누기연산은 왼쪽에서부터 오른쪽순서로
- 더하기와 덜기연산은 왼쪽에서부터 오른쪽순서로

연산순서를 명심하면서 프로그람에서 요구되는 수학연산을 진행하자면 함수들을 잘 리용하여야 한다.

그러면 먼저 표준함수들에 대하여 그 리용방법을 실례를 들면서 설명하자.

2. 표준함수

① Abs()

형식
Abs(수)

Abs함수는 수의 절대값을 돌려주는 표준함수이다.

수인수는 임의의 값을 가지는 수식이 될수 있다.

만일 수인수로 지적된 변수가 초기화 된 경우에는 령을 돌려준다.

[실례 1]

```
STATIC PUBLIC SUB Main()
  DIM MyNum AS Variant
```

```
  DIM x as float
  MyNum = Abs(1.03E3)
  PRINT MyNum
  MyNum = Abs(2.5-6.5)
  PRINT MyNum
END
```

결과

1030

4

0

② ACs()/ACos()

> 형식
> ACs(수)
> ACos(수)

ACs()/ACos()함수는 수의 거꿀코시누스 (arccos)를 계산하는 함수이다.
수의 뜻구역은 [1,1]이다.
수값은 라디안으로 표시해야 하며 계산값도 라디안으로 표시된다.

[실례 2]

```
PRINT Acs(0.5)
1.047197551197

PRINT Acs(1)
3.14159265359
```

③ Acsh()/ACosh()

> 형식
> Acsh(수)
> ACosh(수)

Acsh()/ACosh()는 함수는 수의 거꿀쌍곡코시누스를 계산하는 함수이다.
수인수는 라디안으로 표시하며 계산값은 라디안으로 표시된다.
뜻구역은 X >= 1의 모든 실수값을 포함한다.
값구역은 Y >= 0이다.

[실례 3]

```
STATIC PUBLIC SUB Main()
  DIM y AS Variant
  Y = 2
  PRINT Acsh(Y)
END
```

결과:1.316957896925

④ Asn()/ASin()

> 형식
>
> Asn(수)
>
> ASin(수)

이 함수는 수의 거꿀시누스를 계산하는 함수이다.

뜻구역은 [1,1]으로서 라디안으로 주며 값구역도 라디안으로 표시된다.

[실례 4]

```
STATIC PUBLIC SUB Main()
  DIM y AS Variant
  Y = 0.5
  PRINT Asn(Y)
  Y = -1.0
  PRINT Asn(Y)
END
```

결과

0.523598775598

1.570796326795

⑤ Asnh()/ASinh()

> 형식
>
> Asnh(수)
>
> ASinh(수)

Asnh()/ASinh()함수는 수의 거꿀쌍곡시누스를 계산하는 함수이다.

뜻구역과 값구역의 모든 값들은 라디안으로 표시된다.

[실례 5]

```
STATIC PUBLIC SUB Main()
```

```
    DIM y AS Variant

    Y = 2
    PRINT Asnh(2)
  END
```
결과

1.443635475179

⑥ Atn()/ATan()

> **형식**
>   Atn(수)
>   ATan(수)

Atn()/ATan()함수는 수의 거꿀탕겐스(arctan)값을 계산하는 함수이다.
X의 실수값에 대하여 함수는 [π/2, π/2]범위의 값을 돌려준다.
모든 값은 라디안으로 표시된다.

[실례 6]

```
  STATIC PUBLIC SUB Main()
    DIM y AS Variant

    Y = 0.5
    PRINT ATan(Y)
  END
```
결과

0.463647609001

⑦ Atnh()/ATanh()

> **형식**
>   Atnh(수)
>   ATanh(수)

Atnh()/ATanh()함수는 수의 거꿀쌍곡탕겐스를 계산하는 함수이다.
뜻구역과 값구역의 모든 값들은 라디안으로 표시된다.

[실례 7]

```
  STATIC PUBLIC SUB Main()
    DIM MyResult AS Float
```

```
  DIM MyNum AS Float

  MyNum = 0.5
  MyResult = ATanh(MyNum)
  PRINT MyNum &"의 거꿀쌍곡탕겐스값: " & MyResult
END
```

결과

0.5의 거꿀쌍곡탕겐스값: 0.549306144334

⑧ Cos()

형식 Cos(수)

Cos()함수는 수의 코시누스값을 계산하는 함수이다.
여기서 수는 라디안으로 표시된 각이다.
이 함수의 값구역은 [-1,1]이다.

[실례 8]

```
STATIC PUBLIC SUB Main()
 DIM MyAngle AS Float
 DIM MySecant AS Float
 MyAngle = 1.3                    ' 각을 라디안으로 정의하여 준다.
 MySecant = 1 / Cos(MyAngle)    ' 코쎄칸스계산.
 PRINT  MyAngle &  "라디안의 코쎄칸스값: " & MySecant
END
```

결과

1.3라디안의 코쎄칸스값: 3.738334127075

[주의]

각을 라디안으로 표시하자면 각*π/180.
라디안을 각으로 표시하자면 라디안*180/π.

⑨ Cosh()

형식 Cosh(수)

이 함수는 수의 쌍곡코시누스를 계산하는 함수이다.
모든 값들은 라디안으로 표시된다.

[실례 9]

```
STATIC PUBLIC SUB Main()
 DIM MyAngle AS Float
 DIM MyResult AS Float

 MyAngle = 1.0  ' 라디안으로 표시된 값.
 MyResult = Cosh(MyAngle)
 PRINT MyAngle & "라디안의 쌍곡코시누스값: " & MyResult
END
```

결과

1라디안의 쌍곡코시누스값 : 1.543080634815

⑩ Deg()와 Rad()

형식
Deg(수) Rad(수)

Deg()함수는 라디안을 각으로 변화시키는 함수이다.

Rad()함수는 이와 반대로 각을 라디안으로 변화시키는 함수이다.

[실례 10]

```
STATIC PUBLIC SUB Main()
  PRINT Deg(Pi/2)
  PRINT Rad(90)
END
```

결과

90

1.570796326795

⑪ Exp()

형식
Exp(X)

이 함수는 e의 x제곱(여기서 e는 자연로그의 밑수로서 약 2.718282이다.)을 계산하는 함수이다.

만일 X의 값이 709.782712893을 초과하면 오유가 발생한다.

[실례 11]

```
STATIC PUBLIC SUB Main()
 DIM MyAngle AS Float
 DIM MyHSin AS Float ' 각을 라디안으로 정의.
 MyAngle = 1.3   '지수함수를 리용하여 쌍곡시누스값계산
 MyHSin = (Exp(MyAngle)- Exp(-1 * MyAngle)) / 2
 PRINT MyHSin
END
```

결과

1.698382437293

⑫ Fix()와 Frac()

형식 Fix(수) Frac(수)

Fix()함수는 수의 옹근수부를 돌려주는 함수이다.

Frac()함수는 수의 소수부를 돌려주는 함수이다.

[실례 12]

```
STATIC PUBLIC SUB Main()
 PRINT Fix(Pi)
 PRINT Frac(Pi)
 PRINT Fix(-Pi)
 PRINT Frac(-Pi)
END
```

결과

3

0.14159265359

-3

0.14159265359

⑬ Int()

형식 Int(수)

이 함수는 수의 옹근수부 즉 주어진 수를 넘지 않는 가장 큰 옹근수를 돌려준다.

[실례 13]

```
PRINT Int(Pi)
3
```

⑭ Log()

형식
Log(수)

수는 0보다 큰 임의의 수식이 될수 있다.

이 함수는 밑수 e를 가진 자연로그를 계산하는 함수이다.

상수 e는 약 2.718 282로서 쌍곡선 y = 1/x과 x축, 그리고 수직선 x=1, x=e로 둘러막힌 지역의 면적이 1로 되는 유일한 값이다.

lnx는 물리학과 공학에서 자연로그에 대하여 말할 때 리용된다.

한편 수학자들은 공통적으로 log표시를 사용한다.

[실례 14]

```
STATIC PUBLIC SUB Main()
 PRINT Log(2.71828)
 PRINT Log(1)
END
```

결과

```
0.999999327347
0
```

※ 다음의 공식을 리용하여 임의의 밑수 n에 대한 로그값을 계산할수 있다.

Logn(x) = Log(x)/Log(n)

[실례 15]

```
STATIC PUBLIC SUB Main()
  DIM n AS Float
  DIM x AS Float
  DIM logn AS Float
  n = 2
  x = 10
  logn = Log(x) / Log(n)
  PRINT logn & "은 Log& n & "(" & x & ")" & "의 결과이다. "
END
```

결과

3.321928094887은 Log2(10)의 결과이다.

⑮ Log10()

```
형식
    Log10(수)
```

Log10은 수의 상용로그를 계산하는 함수이다.

즉 Log10(x) = Log(x)/Log(10)

[실례 16]

```
PRINT Log10(10)
1
```

⑯ Max()와 Min()

```
형식
    Max(식 [, 식 ... ])
    Min(식 [, 식 ... ])
```

Max()함수는 식목록에서 값이 가장 큰 식을 돌려준다.

Min()함수는 식목록에서 값이 가장 작은 식을 돌려준다.

식은 수값형 혹은 날자/시간형이 될수 있다.

[실례 17]

```
STATIC PUBLIC SUB Main()
 PRINT Max(16, 4, 7, -1, 3)
 PRINT Max(Now, CDate("01/01/1900"), Cdate("01/01/2100"))
 PRINT
 PRINT Min(16, 4, 7,-1, 3)
 PRINT Min(Now, CDate("01/01/1900"), CDate("01/01/2100"))
END
```

결과

```
16
01/01/2100

-1
01/01/1900
```

⑰ Pi()

> 형식
>
> Pi([수])

Pi()함수는 수*π값을 돌려준다.

만일 수가 지적되지 않으면 1로 간주된다.

[실례 18]

```
PRINT Pi
3.14159265359

PRINT Pi(2)
6.28318530718
```

⑱ Randomize()

> 형식
>
> Randomize(수)

Randomize()함수는 Rnd()함수의 우연수발생을 초기화한다.

즉 이 함수는 자체로 현재의 날자와 시간에 기초하여 Rnd()함수가 우연수를 발생할수 있는 초기값을 결정한다.

만일 Randomize()를 먼저 리용하지 않고 Rnd()함수를 리용하면 프로그람을 실행시킬 때 매번 꼭같은 우연수렬이 발생되게 된다.

같은 프로그람에 대하여 그것을 실행시킬 때마다 새로운 우연수렬이 나타나도록 하려면 Rnd()함수를 호출하기전에 반드시 Randomize()함수를 호출해야 한다.

⑲ Rnd()

> 형식
>
> Rnd([최소값[, 최대값]])

만일 아무런 파라메터도 지적하지 않으면 Rnd()함수는 [0, 1]구간의 우연수를 발생한다.

하나의 파라메터만 지적하면 Rnd()함수는 [0, 최소값]구간의 우연수를 발생한다.
두개의 파라메터를 다 지적하면 Rnd()함수는 [최소값, 최대값]구간의 우연수를 발생한다.

[주의]

임의의 주어진 구간에 속하는 우연옹근수를 만들자면 다음의 공식을 리용하여야 한다.

Int((최대값- 최소값+ 1) * Rnd + 최소값)

[실례 19]

```
STATIC PUBLIC SUB Main()
 DIM Dice AS Integer

 PRINT "0 과 1 사이값: ";
 PRINT Rnd
 PRINT "0 과 2 사이값: ";
 PRINT Rnd(2)
 PRINT "Pi와 Pi*2 사이값: ";
 PRINT Rnd(Pi, Pi(2))
 PRINT
 Randomize

 DO WHILE Dice <> 1
  Dice = Int(Rnd(1,7))
  '주사위던지기를 모의하여 1부터 6까지 우연수를 발생한다.
  PRINT "던지기" & dice
 LOOP
END
```

결과

```
0 과 1 사이값:
7.826369255781E-6
0 과 2 사이값:
0.263075576164
Pi와 Pi*2 사이값:
5.515396781706

던지기 6
던지기 4
던지기 1
```

⑳ Round()

형식
Round(수[, 자리수])

Round()함수는 둥그리기함수이다.

자리수가 지적되지 않는 경우 옹근수에 가장 가까운 수로 둥그리기한다. 자리수를 지적하면 10^자리수값으로 둥그리기한다.

[실례 20]

```
STATIC PUBLIC SUB Main()
 PRINT Round(Pi)
 PRINT Round(Pi, -2)
 PRINT Round(1972, 2)
END
```

결과

```
3
3.14
2000
```

㉑ Sgn()

형식
Sgn(수)

Sgn()함수는 수의 부호를 지적하는 옹근수를 돌려준다.

만일 수값이 령이면 0을 돌려주고 수값이 정수이면 +1을 돌려준다. 수값이 부수이면 -1값을 돌려준다.

[실례 21]

```
STATIC PUBLIC SUB Main()
 PRINT Sgn(Pi)
 PRINT Sgn(-Pi)
 PRINT Sgn(0)
END
```

결과

```
1
-1
0
```

㉒ Sin()

형식
Sin(각)

Sin()함수는 각의 시누스값을 계산하는 함수이다.
각은 라디안으로 준다.

[실례 22]

```
STATIC PUBLIC SUB Main()
 PRINT Sin(Pi/2)
END
```

결과

1

⑳ Sinh()

형식 Sinh(수)

Sinh()함수는 수의 쌍곡시누스를 계산하는 함수이다.

[실례 23]

```
STATIC PUBLIC SUB Main()
 DIM MyAngle AS Float
 DIM MyHSin AS Float    '각을 라디안으로 정의

 MyAngle = 1.3    ' 지수함수를 리용하여 쌍곡시누스계산
 MyHSin = (Exp(MyAngle) - Exp(-1 * MyAngle)) / 2
 PRINT MyHSin
 PRINT Sinh(MyAngle)' 해당함수를 리용하여 계산
END
```

결과

1.698382437293
1.698382437293

㉔ Sqr()

형식 Sqr(수)

Sqr는 수의 두제곱뿌리를 계산하는 함수이다.
수인수는 부아닌 수값을 가지는 수식이다.

[실례 24]

```
STATIC PUBLIC SUB Main()
 DIM MySqr AS Float

 MySqr = Sqr(4)
 PRINT MySqr & ":Sqr(4)의 결과"
 MySqr = Sqr(23)
 PRINT MySqr & ":Sqr(23)의 결과"
 MySqr = Sqr(0)
 PRINT MySqr & ":Sqr(0)의 결과"
END
```

결과

2: Sqr(4)의 결과

4.795831523313 :Sqr(23)의 결과

0: Sqr(0)의 결과

㉕ Tan()

형식
Tan(각)

Tan()함수는 각에 대한 탕겐스값을 계산하는 함수이다.

각은 라디안값을 주는 임의의 수식이 될수 있다.

[실례 25]

```
STATIC PUBLIC SUB Main()
 DIM MyAngle AS Float
 DIM MyCotangent AS Float

 MyAngle = 1.3   '각을 라디안으로 정의.
 MyCotangent = 1/Tan(MyAngle)   ' 코탕겐스계산.
 PRINT MyCotangent
 PRINT Tan(Pi/4)
END
```

결과

0.277615646541

1

㉖ Tanh()

형식

```
Tanh(수)
```

Tanh()함수는 수의 쌍곡탕겐스값을 계산하는 함수이다.

[실례 26]

```
STATIC PUBLIC SUB Main()
 PRINT Tanh(1)
END
```

결과

```
0.761594155956
```

3. 유도함수와 그 적용방법

수학계산에서 리용되는 다른 많은 함수들은 우의 표준함수들을 리용하여 만들수 있다. 다음의 표에서는 수학공식을 리용하여 만들수 있는 간단한 유도함수들을 보여준다.

표 5-1. 유 도 함 수 들

함 수	공 식
쎄칸스	Sec(X) = 1 / Cos(X)
코쎄칸스	Cosec(X) = 1 / Sin(X)
코탕겐스	Cotan(X) = 1 / Tan(X)
거꿀쎄칸스	Arcsec(X)=Atn(X/Sqr(X*X-1))+Sgn((X)-1)*(2*Atn(1))
거꿀코쎄칸스	Arccosec(X)=Atn(X/Sqr(X*X - 1))+(Sgn(X) - 1)*(2*Atn(1))
거꿀코탕켄스	Arccotan(X) = Atn(X) + 2 * Atn(1)
쌍곡쎄칸스	HSec(X) = 2 / (Exp(X) + Exp(-X))
쌍곡코세칸스	HCosec(X) = 2 / (Exp(X) - Exp(-X))
쌍곡코탕켄스	HCotan(X)=(Exp(X)+Exp(-X))/(Exp(X)- Exp(-X))
거꿀쌍곡쎄칸스	HArcsec(X) = Log((Sqr(-X * X + 1) + 1) / X)
거꿀쌍곡코쎄칸스	HArccosec(X)=Log((Sgn(X)*Sqr(X*X+1)+1)/X)
거꿀쌍곡코탕겐스	HArccotan(X) = Log((X + 1) / (X - 1)) / 2
밑수 n을 가지는 로그	LogN(X) = Log(X) / Log(N)

Gambas는 이와 같은 유도함수들을 직접 제공하지 못하기때문에 이 장의 마지막 련습으로 우의 표에 렬거된 함수들을 제공할수 있도록 능력을 확장하는 모듈을 작성하려고 한다.

작성하는 모듈의 이름은 Derived이며 이 모듈이 완성되면 사용자들은 **《Derived.함수이름》**형식으로 유도함수들을 리용할수 있다.

Gambas를 기동하고 MyMath라는 이름을 가진 조작탁응용프로그람을 창조하시오.

그리고 새로운 클라스로서 Class1을 창조하고 그것을 시작클라스로 하시오.

다음 새로운 모듈을 창조하고 그 이름은 Derived라고 하시오.

Class1의 코드창을 열고 다음과 같이 코드를 작성한다.

```
'Gambas class file
  STATIC PUBLIC SUB Main()
   DIM my AS Float

   my = 51.5
   WHILE my <> 0.0
    PRINT "1보다 큰수를 입력하시오. 끝내려면 0을 입력"
    LINE INPUT my
    IF my = 0.0 THEN BREAK
    IF my <= 1.0 THEN
     PRINT "입력값이 1보다 작다."
           CONTINUE
    ENDIF
    PRINT "입력값: " & my
    PRINT Derived.Sec(my) & ":secant"
    PRINT Derived.CoSec(my) & ":cosecant"
    PRINT Derived.CoTan(my) & ":cotangent"
    PRINT
    PRINT Derived.ArcSec(my) & ":arcsecant"
    PRINT Derived.Arccosec(my) & ":arccosecant"
    PRINT Derived.Arccotan(my) & ":arccotangent"
    PRINT
    PRINT Derived.HSec(my) & ":hyperbolic secant"
    PRINT Derived.HCoSec(my) & ":hyperbolic cosecant"
    PRINT Derived.HCoTan(my) & ":hyperbolic cotangent"
```

```
    PRINT
    PRINT Derived.HArcSec(my) & ":hyperbolic arcsecant"
    PRINT Derived.HArcCoSec(my) & ":hyperbolic arccosecant"
    PRINT Derived.HArcCoTan(my) & ":hyperbolic arccotan"
    PRINT
    PRINT Derived.LogN(my,2)&":"&myVal&"의 밑수 2인 로그값"
   WEND
  END
```

다음은 Derived 모듈파일을 두번 찰칵하여 열고 다음의 코드를 넣으시오.

```
' Gambas Derived.module file
  PUBLIC FUNCTION Sec(x AS Float) AS Float
   DIM result AS Float
   result = 1.0/Cos(x)
   RETURN result
  END

  PUBLIC FUNCTION CoSec(x AS Float) AS Float
   DIM result AS Float
   result = 1.0/Sin(x)
   RETURN result
  END

  PUBLIC FUNCTION CoTan(x AS Float) AS Float
   DIM result AS Float
   result = 1.0/Tan(x)
   RETURN result
  END

  PUBLIC FUNCTION ArcSec(x AS Float) AS Float
   DIM result AS Float
   result = Atn(x/Sqr(x*x-1))+Sgn((x)-1)*(2*Atn(1))
   RETURN result
  END
```

```
PUBLIC FUNCTION Arccosec(x AS Float) AS Float
 DIM result AS Float
result = Atn(x/Sqr(x*x-1))+(Sgn(x)-1)*(2*Atn(1))
 RETURN result
END

PUBLIC FUNCTION Arccotan(x AS Float) AS Float
 DIM result AS Float
 result = Atn(x)+(2*Atn(1))
 RETURN result
END

PUBLIC FUNCTION HSec(x AS Float) AS Float
 DIM result AS Float
 result = 2.0/(Exp(x)+Exp(-x))
 RETURN result
END

PUBLIC FUNCTION HCoSec(x AS Float) AS Float
 DIM result AS Float
 result = 2.0 / (Exp(x) - Exp(-x))
 RETURN result
END

PUBLIC FUNCTION HCoTan(x AS Float) AS Float
 DIM result AS Float
 result = (Exp(x) + Exp(-x))/(Exp(x) - Exp(-x) )
 RETURN result
END
'만일 공식 Log((Sqr(x*x+1))+1)/x)를 그대로 사용하면
' Sqr함수와 Log함수는 부수값을 가질 때 오유가 발생할수 있다.
'부분수속을 다음과 같이 작성하면 오유없이 같은 결과를 줄수 있다.
PUBLIC FUNCTION HArcsec(x AS Float) AS Float
 DIM result AS Float
```

```
 DIM Temp1 AS Float
 DIM Temp2 AS Float
 DIM Temp3 AS Float
 result = Temp3 * 1

 Temp1 = ((x * -1)*( x + 1))+1
 Temp2 = Sqr(Abs(Temp1))
 Temp3 = Log(Temp2)
 result = Temp3 *- 1

 RETURN result
END

PUBLIC FUNCTION HArccosec(x AS Float) AS Float
 DIM result AS Float
 result = Log((Sgn(x)*Sqr(x*x+1)+1)/x)
 RETURN result
END

PUBLIC FUNCTION HArccotan(x AS Float) AS Float
 DIM result AS Float
 result = Log((x+1)/(x-1))/2
 RETURN result
END

PUBLIC FUNCTION LogN(x AS Float, n AS Float) AS Float
 DIM result AS Float
 result = Log(x)/Log(n)
 RETURN result
END
```

프로그람을 실행시키면 《1보다 큰수를 입력하시오. 끝내려면 0을 입력》라는 통보가 나온다.

그때 4를 입력하면 다음의 결과가 나온다.

```
1보다 큰수를 입력하시오. 끝내려면 0을 입력
```

```
입력값: 4
1.529885656466 :secant
1.321348708811 :cosecant
0.863691154451 :cotangent

0.85707194785 :arcsecant
0.801529994232 :arccosecant
2.896613990463 :arccotangent

0.036618993474 :hyperbolic secant
0.036643570326 :hyperbolic cosecant
1.000671150402 :hyperbolic cotangent

1.472219489583 :hyperbolic arcsecant
0.247466461547 :hyperbolic arccosecant
0.255412811883 :hyperbolic arccotangent
2 : 4 의 밑수 2인 로그값
1보다 큰수를 입력하시오. 끝내려면 0을 입력
```

이와 같은 방법으로 프로그람에서 필요한 거의 모든 수학연산을 진행할수 있게 된다.

제 6 장. 도형사용자대면부프로그람작성

지금까지는 간단한 조작탁형프로그람들만을 작성하였다.

실천에서는 사용자들이 대면부를 가진 프로그람을 요구한다. 프로그람에서 대면부는 항상 눈에 띄우는것이기때문에 제일 중요한 부분이라고 할수 있다.

도형사용자대면부를 가진 프로그람을 작성하려면 대면부를 잘 설계하고 작성하여야 한다.

이 장에서는 먼저 대면부작성도구인 콘트롤의 개념과 종류, 속성설정방법에 대하여 학습한다. 다음 메쏘드와 사건들에 대하여 학습하고 이것을 종합하여 도형사용자대면부프로그람을 작성하는 방법을 실례프로그람을 통하여 련습하게 된다.

1. 콘트롤

도형사용자대면부를 작성하자면 도구창에 있는 콘트롤들을 홈설계창문우에 배치하여야 한다.

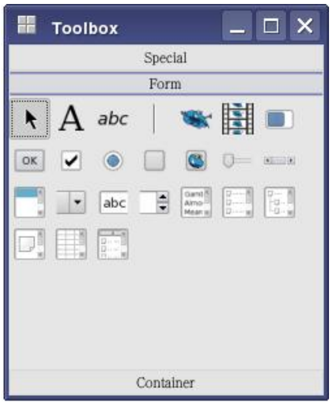


그림 6-1. Gambas 도구창

그림 6-1에서 보는바와 같이 도구창은 Form과 Container, Spacial 세개의 부분으로 구성되여있다.

Gambas를 기동하면 항상 도구창에 표준적으로 나타나는 콘트롤들이 있다.

이 콘트롤들을 표준콘트롤이라고 한다.

① **표준콘트롤**

표준콘트롤들은 세개의 부분도구창에 나누어져있다.

- *Form부분도구창에 있는 콘트롤들*

그림 6-1에 표시된 콘트롤들이 Form부분도구창에 소속된 콘트롤들이다.

● 왼쪽우에서부터 첫번째행에는 Select(선택), Label(표식), TextLabel(본문표식), Separater(분리), PictureBox(그림칸), MovieBox(동영상칸), ProgressBar(진행띠)콘트롤들이 있으며

● 두번째행에는 Button(단추), CheckBox(검사칸), RadioButton(단일선택단추), ToggleButton(토굴단추), ToolButton(도구단추), Slider(미끄럼띠), ScrollBar(흘림띠)콘트롤이 있다.

● 세번째행에는 ListBox(목록칸), ComboBox(결합칸), TextBox(본문칸), SpinBox(스핀칸), TextArea(본문령역), ListView(목록구조), TreeView(나무구조)콘트롤이 있으며

● 네번째행에는 IconView(아이콘구조), GridView(살창구조), ColumnView(렬구조)콘트롤들이 있다. 다른 부분도구창은 도구창에서 해당한 부분도구창이름을 마우스로 찰칵하여 펼칠수 있다.

- container부분도구창에 있는 콘트롤들

그림 6-2에서 보는바와 같이 첫번째행에는 Select(선택), HBox(수평칸), VBox(수직칸), HPanel(수평판), VPanel(수직판), Hsplit(수평분리), Vsplit(수직분리)콘트롤이 있으며 두번째행에는 Panel(판), Frame(틀), TabStrip(타브판), ScrollView(흘림띠구조), DrawingArea(그리기령역)콘트롤이 있다.

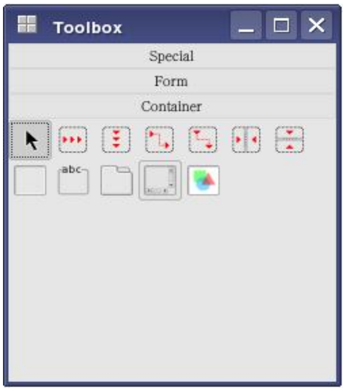


그림 6-2. container부분도구창

- Special부분도구창에 있는 콘트롤들

여기에는 Select(선택), Timmer(시계콘트롤), Embedder(매몰), Tray icon(생각아이콘)콘트롤이 있다.



그림 6-3. special부분도구창

표준콘트롤외에 다른 외부콘트롤들을 사용하자면 Project/ Properties를 찰칵하고 components를 선택한 다음 해당 콘트롤묶음을 지적하면 된다.

보는바와 같이 Gambas에는 많은 콘트롤들이 있다.

그러면 대면부에서 가장 많이 리용되는 콘트롤인 단추콘트롤(Button)을 통하여 콘트롤을 창조하는 방법부터 먼저 학습한다.

② 콘트롤의 창조

단추콘트롤을 리용하면 홈에 무엇인가 <확인>하거나 <중지>하는것과 같은 단추를 완성할수 있다.

단추우에는 본문 또는 그림도 표시할수 있다.

물론 도구창에서 단추아이콘을 선택하여 홈에 단추콘트롤을 배치할수 있다.

이러한 객체를 홈우에 창조할수 있다는것은 곧 실행시에도 홈에 동적으로 단추를 발생시키는 코드를 작성할수 있다는것을 의미한다.

단추객체를 선언하자면 다음과 같은 형식을 리용해야 한다.

```
DIM hButton AS Button
hButton = NEW Button(Parent AS Container)
```

먼저 Button형변수로서 hButton을 선언한 다음 NEW에 의하여 hButton에 객체를 창조한다.

여기서 Container는 다른 콘트롤들을 포함하는 어미클라스로서 이 경우에는 단추가 배치되는 홈을 의미한다.

우의 코드에 의하여 프로그람실행시에 홈에는 새로운 단추가 동적으로 창조된다.

콘트롤을 창조한 다음에는 콘트롤의 형태를 자기마음대로 만들어야 하는데 이것은 콘트롤의 속성에 대한 설정과 조종을 통하여 진행한다.

2. 속성

콘트롤의 속성은 대면부설계시에 속성창을 리용하여 직접 설정할수도 있고 프로그람의 실행시에 설정하거나 읽을수도 있다.

실례로 다음 코드는 단추의 본문속성을 설정하여 단추우에 <OK>가 표시되게 한다.

```
hButton.Text = "OK"
```

콘트롤의 속성을 설정하는 명령문은 《콘트롤이름.속성=식》의 형식을 가진다. 이 형식은 콘트롤에서 속성정보를 얻거나 설정할 때 리용되는 표준적인 방법이다.

모든 콘트롤들에서 대부분 속성은 공통적이다.

따라서 여기서는 공통적인 속성들을 먼저 학습하고 나중에 주어진 콘트롤들에서 특이한 속성들을 학습하게 된다.

① Name

콘트롤의 이름속성이다. 대면부설계시에 콘트롤을 배치하면 콘트롤의 이름은 표준적으로 설정된다. 실례로 Button콘트롤에서는 보통 button1, button2 등의 형태로 이름이 설정된다.

그러나 속성창에서 이 속성을 변화시켜 사용자의 요구에 맞게 콘트롤의 이름을 변경할수 있다.

실례로 Quitbtn, Openbutton…

이 속성은 대면부의 설계시에만 변화시킬수 있고 실행시에는 변화시킬수 없다.

② BackColor

콘트롤의 배경색속성으로서 옹근수값으로 설정한다.

색값을 설정하기 위하여 미리 정해진 색갈상수값을 리용할수 있다.

Black	Blue	Cyan	DarkBlue
DarkCyan	DarkGray	DarkGreen	DarkMagenta
DarkRed	DarkYellow	Default	Gray
Green	LightGray	Magenta	Orange
Pink	Red	Transparent	Violet
White	Yellow		

실례로 hButton의 배경색을 붉은색으로 설정하자면 hButton.BackColor = Color.Red와 같이 설정하면 된다.

만일 설정하려는 색에 대하여 RGB 혹은 HSV값을 알면 color클라스의 RGB()와 HSV()메쏘드를 리용하여 색값으로 변환하여 배경속성에 줄수 있다.

- *RGB()의 형식*

RGB(붉은색값, 풀색값, 푸른색값)

RGB는 붉은색, 풀색, 푸른색값을 0~255사이의 옹근수로 설정하여주며 그에 해당한 색값을 돌려준다.

- *HSV()의 형식*

HSV(색상, 투명도, 명암)

HSV는 색상, 투명도, 명암(밝기도)요소들로 색값을 돌려준다.

[실례 1]

```
hButton.BackColor = Color.RGB(255,255,255)
```

이 코드는 단추의 배경색을 흰색으로 설정한다.

RGB()와 HSV()를 리용하면 Gambas가 제공하는 고정적인 색갈상수값외에 다른 색값들을 설정하려고 할 때 유리하다.

③ Border

콘트롤의 경계를 설정하는 속성으로서 많은 콘트롤들에서 찾아볼수 있는 공통적인 속성이다.

경계속성설정에 리용되는 상수들은 다음과 같다.

Etched, None, Plain, Raised, Sunken

[실례 2]

```
hButton.Border = Border.Plain
```

④ Cancel

이것은 사용자가 건반에서 ESC건을 눌렀을 때 마치도 해당 단추를 마우스로 찰칵한것처럼 동작하겠는가 안하겠는가를 결정하는 속성이다.

이 속성은 론리값으로 설정된다.

[실례 3]

```
hButton.Cancel = TRUE
```

프로그람이나 속성창에서 단추의 Cancel속성을 True로 설정하면 마치도 사용자가 <중지>, <탈퇴>, <Cancel>과 같은 단추를 누른것처럼 사건을 조종하여 홈에서 유연하게 탈퇴할수 있게 해준다.

⑤ Caption/Text

문자렬값을 가지는 속성으로서 콘트롤에 문자렬을 표시하거나 돌려주는 속성이다.

[실례 4]

```
hButton.Caption = "OK"
hButton.Text = "OK"
```

⑥ Default

이것은 사용자가 건반에서 ENTER건을 눌렀을 때 마치도 해당 단추를 마우스로 찰칵한것처럼 동작하겠는가 안하겠는가를 결정하는 속성이다.

이 속성은 론리값으로 설정된다.

일반적으로 <확인>/<중지>형의 대화창을 창조할 때 이 속성을 사용하여 단추들중의 하나를 고정동작으로 설정하게 된다.

[실례 5]

```
hButton.Default = FALSE
```

⑦ DROP

이것은 콘트롤의 끌기와 놓기로부터 놓기를 받아들이겠는가를 결정하거나 검사하는데 리용되는 속성이다.

아래와 같은 코드를 리용하여 콘트롤우에 무엇인가를 놓기하기전에 값을 검증할수 있다.

[실례 6]

```
IF hButton.Drop = FALSE THEN
   hButton.ToolTip = "놓기가 허용되지 않는다"
ELSE
   hButton.ToolTip = "여기에는 놓기가 허용된다."
ENDIF
```

⑧ Enabled

이속성은 콘트롤을 리용할수 있는가 없는가를 지적한다.

콘트롤을 동작할수 없게 하자면 이 속성값을 FALSE로 한다.

[실례 7]

```
IF 어떤조건식 = FALSE THEN
   hButton.Enabled = TRUE
ELSE
   hButton.Enabled = FALSE
ENDIF
```

⑨ FONT

이 속성은 콘트롤에서 본문을 나타내기 위하여 사용되는 서체를 돌려주거나 설정하여준다.

[실례 8]

```
hButton.Font.Name = "PRK P Chonbong"
hButton.Font.Bold = TRUE
hButton.Font.Italic = FALSE
hButton.Font.Size = "10"
hButton.Font.StrikeOut = FALSE
hButton.Font.Underline = FALSE
```

결과



그림 6-4. 폰트속성설정결과

⑩ Forecolor

이 속성은 콘트롤의 전경색을 설정해준다.

Foreground속성도 ForeColor속성과 같다.

이 속성에 값을 설정할 때에도 색갈상수값을 리용할수 있다

[실례 9]

```
hButton.ForeColor = Color.Red
hButton.Foreground = Color.RGB(255,255,255)
```

⑪ X

이 속성은 옹근수로 정의되는 속성으로서 콘트롤의 가로방향위치값을 설정하거나 돌려주는데 리용된다.

Left속성도 X속성과 같다.

⑫ Y

이 속성도 옹근수로 정의되는 속성으로서 콘트롤의 세로방향위치를 설정하거나 돌려주는데 리용된다.

⑬ Height

이 속성은 옹근수로 정의되는 속성으로서 콘드롤의 높이를 설정하거나 돌려주는데 리용된다. 이것은 H속성과 류사하며 교환하여 쓸수 있다.

⑭ Width

이 속성은 옹근수로 정의되는 속성으로서 콘트롤의 너비를 설정하거나 돌려주는 속성이다.

이와 류사하게 W속성이 있으며 서로 교환하여 쓸수 있다.

⑮ Mouse

이 속성은 옹근수로 정의되는 속성으로서 마우스가 콘트롤의 경계내부에서 움직일 때 마우스지시자의 형태를 설정하거나 돌려주는 속성이다.

마우스지시자의 형태	값
Default	1
Arrow	0
Cross	2
Wait	3
Text	4
SizeS	5
SizeE	6
SizeNESW	7
SizeNWSE	8
SizeAll	9
Blank	10
SplitV	11
SplitH	12
Pointing	13

⑯ Parent

이 속성은 읽기전용속성으로서 콘트롤을 포함하는 용기(Container)를 돌려주는데리용된다.

⑰ Picture

이 속성은 콘트롤우에 보여주는 그림을 설정하거나 돌려주는 속성이다.

⑱ ScreenX

이것은 화면좌표에서 콘트롤의 왼쪽경계위치를 돌려주는 읽기전용속성이다.

보통 ScreenY속성도 함께 사용된다.

⑲ ScreenY

이것은 화면좌표에서 콘트롤의 웃경계위치를 돌려주는 읽기속성이다.

⑳ ToolTip

이 속성은 문자렬로 정의되는 속성으로서 ToolTip를 돌려주거나 설정한다.(ToolTip는 마우스를 얼마동안 그 콘트롤우에 가져갈 때 나타나는 자그마한 본문칸이다.)

[실례 10]

```
hButton.ToolTip = "Click me!"
```

㉑ Top

이 속성은 옹근수로 정의되는 속성으로서 콘트롤이 배치되여있는 용기안에서 콘트롤의 웃경계의 위치를 돌려주거나 설정하여준다.

㉒ Visible

이 속성은 론리값으로 정의되는 속성으로서 콘트롤이 보이게 하겠는가 보이지 않게 하겠는가를 결정한다. 이 속성을 FALSE로 설정하면 콘트롤이 보이지 않는다.

반대로 TRUE로 설정하면 콘트롤이 보인다.

3. 메쏘드

메쏘드는 콘트롤에 대하여 그것을 보여주거나 숨기거나 움직이는것 등과 같이 그 무엇인가를 하도록 하는 고정된 부분프로그람이다.

실례로 단추콘트롤에는 Delete, Drag, Grab, Hide, Lower, Move, Raise, Refresh, Resize, SetFocus, Show와 같은 메쏘드들이 있다.

메쏘드를 리용하자면 콘트롤이름.메쏘드(인수1, 인수2, …)의 형식을 리용해야 한다.

메쏘드는 함수처럼 호출한 다음 값을 돌려주는것도 있고 일반 부분프로그람처럼 호출하여 리용만 하는것이 있다.

① Delete

이 메쏘드는 콘트롤을 없애버린다.

[실례 11]

```
button1.Delete
```

콘트롤을 없애면 그 어떤 코드를 가지고도 그 콘트롤을 다시 살릴수 없다는것을 알고 심중히 리용해야 한다.

② Drag

형식
Drag(Data AS Variant [, Format AS String])

이 메쏘드가 실행되면 해당 콘트롤의 끌기와 놓기공정을 시작한다.

여기서 Data는 끌게 될 자료이다.

그 자료는 문자렬 혹은 화상일수 있다.

③ Grab

형식
Grab() AS Picture

이것은 콘트롤의 그림을 돌려주는 메쏘드이다.

④ Hide와 Show

이 메쏘드들은 콘트롤을 간단히 숨기거나 보이게 한다.

[실례 12]

```
IF 조건식 = TRUE THEN
```

```
    hButton.Hide
  ELSE
    hButton.Show
  ENDIF
```

⑤ Move

형식 Move(X, Y [,너비, 높이])

콘트롤을 움직이거나 그 크기를 변경시키는데 리용된다.

인수값들은 물론 옹근수값이다.

[실례 13]

```
hButton.Move(hButton.X 50, hButton.Y 20)
```

⑥ Refresh

형식 Refresh([X, Y, 너비, 높이])

콘트롤을 다시 그리거나 일부 실체들에서 콘트롤의 위치를 수정하는데 리용된다.

⑦ Resize

형식 Resize(너비, 높이)

콘트롤의 크기를 변경시키는데 리용된다.

⑧ SetFocus

콘트롤에 초점을 주려고 할 때 이 메쏘드를 호출할수 있다.

콘트롤이 초점을 가지면 보통 다른 콘트롤과 구별되여 나타난다.



그림 6-5. 초점을 가진 콘트롤

실례로 초점을 가진 단추에는 그 주위에 작은 점선테두리가 나타난다. 콘트롤이 초점을 가진다는것은 대면부에서 그 콘트롤과의 작업이 우선시 된다는것을 의미한다.

실례로 만일 본문콘트롤이 초점을 가지면 그 콘트롤을 마우스로 지적하지 않고도 건반으로 즉시 자료를 입력할수 있으며 단추콘트롤이 초점을 가지면 ENTER건을 눌러도 마우스로 그 단추를 누른것과 같은 효과가 나타난다.

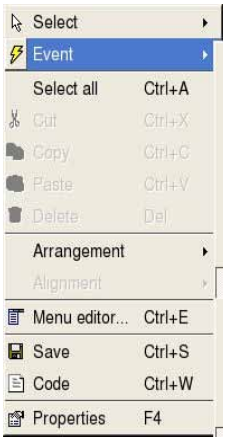


그림 6-6. 사건안내

4. 사건

사건은 사용자나 조작체계가 프로그람(혹은 오브젝트)에 대하여 진행하는 그 어떤 작용을 통털어 이르는 말이다.

실례로 《단추를 누른다》, 《창문을 연다》, 《프로그람이 적재된다》와 같은것을 사건이라고 말할수 있다.

도형사용자대면부프로그람에서는 사건이 프로그람동작의 계기로 된다.

다시말하여 사건이 발생하지 않으면 프로그람은 동작하지 않는다.

프로그람작성자는 매개 사건에 대하여 그것을 처리하기 위한 일련의 코드를 작성하는데 이 코드묶음을 사건수속(Event procedure)이라고 한다.

프로그람작성자가 처리하는 매 사건수속은 객체이름과 사건에 의하여 구분된다.

코드창에서는 사건수속틀거리가 자동적으로 만들어진다.

실례로 단추를 눌렀을 때의 사건수속을 처리하려면 홈에서 단추를 선택하고 오른쪽마우스단추를 찰칵하여 나타나는 안내문에서 Event를 선택한다.

다음 해당한 사건을 선택한다.

이때 선택된 사건수속틀거리는 다음과 같이 코드창에 나타난다.

```
PUBLIC SUB hButton_Click()

END
```

그러면 실례프로그람을 작성하는 과정을 통하여 어떤 사건이 발생하며 그것을 어떻게 처리하는가를 배우고 여러가지 사건처리수법들을 숙련해보자.

[실습 1]

Gambas를 기동하고 PRO1이라는 도형사용자대면부방식의 프로젝트를 창조하시오.

나무구조에서 기본홈을 마우스로 두번 찰칵하면 빈 홈이 열려진다.

빈 홈의 임의의 위치에서 마우스를 두번 찰칵하면 코드창문에 홈이 열릴 때의 사건을 처리하는 사건수속틀거리가 나타난다.

```
' Gambas class file
 PUBLIC SUB Form_Open()

 END
```

여기에 아래와 같은 코드를 입력하시오.

```
 PUBLIC SUB Form_Open()
  DIM hButton AS Button
  hButton = NEW Button (ME) AS "hButton"

  hButton.X = 215
```

```
    hButton.Y = 60
    hButton.Width = 200
    hButton.Height = 40
    hButton.Enabled = TRUE
    hButton.Text = " Click or Press ESC to Quit."
    hButton.Border = TRUE
    hButton.Default = TRUE
    hButton.Cancel = TRUE
    hButton.Show
  END
```

유능한 프로그람작성자는 코드를 작성할 때 수값을 직접 리용하지 않고 미리 정해진 상수들을 항상 리용해야 한다.

실례로 건반코드값들과 마우스형태들, 색갈, 폰트값과 같은것을 정의하는 상수이름들을 잘 알고 그대로 리용하는것이 좋다.

계속하여 단추를 찰칵하였을 때의 사건수속을 완성한다.

```
PUBLIC SUB hButton_Click()
  me.Close
END
```

결국 이 프로젝트에는 두개의 사건 즉 《홈이 열릴 때의 사건》과 《단추를 눌렀을 때의 사건》들을 처리하는 사건수속이 있다. 우의 코드를 작성하고 도구띠에서 풀색단추(실행단추)를 찰칵하면 다음과 같은 결과를 볼수 있다.

첫 사건수속에 의하여 홈이 열리는 순간 단추의 여러가지 속성이 설정되고 마지막으로 Show메쏘드에 의하여 홈에 그림과 같이 단추가 출현한다.



그림 6-7. 첫 프로젝트결과

다음 두번째 사건수속에 의하여 마우스로 단추를 찰칵하거나 건반에서 ESC건을 누르면 이 프로그람환경에서 탈퇴할수 있다. 결국 GUI에 기초한 첫 Gambas프로그람을 개발한 셈이다. 물론 우의 프로그람은 도구창을 리용하면 더 간단히 작성할수 있다.

이미 작성한 프로젝트는 보관하고 새로운 프로젝트를 창조한 다음 홈우에 단추콘트롤을 배치하고 속성창에서 해당한 속성값을 미리 설정한다.

다음 단추콘트롤을 두번 찰칵하면 코드창에는 단추를 찰칵하였을 때의 사건수속부분이 나타난다.

```
PUBLIC SUB hButton_Click()

END
```

여기에 다음의 코드를 작성하여 넣는다.

```
Me.Close
```

그 다음 이 프로그람을 실행시키면 처음과 꼭같이 동작한다.

처음에 Gambas도구창의 우점대신에 코드를 리용하여 힘든 방법으로 프로그람을 개발한것은 다만 프로그람작성자의 코드작성능력을 키우기 위해서이다.

다음은 여러개의 콘트롤로 이루어진 복잡한 형태의 대면부를 가진 프로그람을 작성하여보자.

[실습 2]

Gambas를 다시 기동시킨 다음 프로젝트조수를 리용하여 도형사용자대면부프로젝트를 창조한다. 통합개발환경이 나타나면 이름이 Form1이라는 새로운 시작클라스홈을 창조한다.

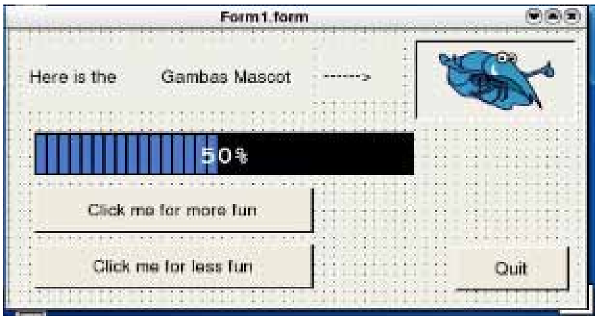


그림 6-8. Form1의 설계

우리가 작성하려는 프로젝트의 대면부는 그림 6-8과 같다.

먼저 그림 6-8과 같이 홈에 3개의 표식콘트롤을 배치한다. 표식콘트롤들의 이름은 각각 Label1, Label2 , Label3으로 한다. 다음 속성창을 열고 Label1의 text속성을

"Here is the"로, Label2의 text속성을 "Gambas Mascot"로, Label3의 text속성을 "----->"로 설정한다.

다음 콤퓨터에서 "gambas mascot.png"로 된 푸른 새우그림을 골라내여 그것을 프로젝트등록부의 Data목록에 복사하여 넣는다.

다음 PictureBox1의 picture속성을 "gambas mascot.png"로 설정한다.

그러면 홈의 웃부분은 그림 6-9와 같이 된다.



그림 6-9. 4개의 콘트롤로 구성된 홈

그림 6-8을 참고하면서 진행띠와 3개의 단추를 홈에 배치한다.

단추들의 text속성을 FunBtn, NoFunBtn, QuitBtn라고 입력한다.

이것으로 홈설계는 끝낸다.

홈설계가 끝나면 프로그람을 동작시키기기 위한 사건처리를 진행한다.

- *홈이 열릴 때의 사건수속*

```
PUBLIC SUB Form_Open()
   ProgressBar1.Value = 0.50
   ProgressBar1.BackColor = Black
   ProgressBar1.ForeColor = Yellow
END
```

이 사건수속에서는 보통 프로그람의 초기화가 진행된다.

여기서는 상태띠의 값을 절반값으로 설정하고 배경색과 전경색을 설정한다.

- *마우스지적자가 홈의 내부로 들어갈때의 사건수속*

```
PUBLIC SUB Form_Enter()
  Label1.Text = ""
  Label2.Text = ""
  Label3.Text = ""
END
```

이것으로 마우스가 홈우를 지나갈 때 표식콘트롤의 본문들이 나타나지 않게 된다.

- *마우스지적자가 홈의 바깥으로 나올때의 사건수속*

```
PUBLIC SUB Form_Leave()
  ProgressBar1.Value = 0.50
```

```
  Label1.Text = "Wasn't that"
  Label2.Text = "some real"
  Label3.Text = "fun stuff?"
END
```

- *마우스지적자가 세개의 표식콘트롤에 다가갈 때의 사건수속*

```
PUBLIC SUB Label1_Enter()
  Label1.Text = "Here's the"
END

PUBLIC SUB Label2_Enter()
  Label2.Text = "Gambas Mascot"
END

PUBLIC SUB Label3_Enter()
  Label3.Text = "----->"
END
```

- *마우스지적자가 푸른 새우그림에로 다가갈 때의 사건수속*

```
PUBLIC SUB PictureBox1_Enter()
  Label1.Text = ""
  Label2.Text = ""
  Label3.Text = ""
  PictureBox1.Border = Border.Plain
END
```

- *마우스지적자가 푸른 새우그림에서 떠나갈 때의 사건수속*

```
PUBLIC SUB PictureBox1_Leave()
  Label1.Text = "Wasn't that"
  Label2.Text = "some real"
  Label3.Text = "fun stuff?"
  PictureBox1.Border = Border.Sunken
END
```

- *Quit단추를 찰칵하였을 때의 사건수속*

```
PUBLIC SUB QuitBtn_Click()
  Form1.Close
END
```

다음 FunBtn을 찰칵할 때에는 상태띠값이 증가하고 NoFunBtn을 찰칵할 때에는 상태띠값이 감소하게 하려고 한다.

- *FunBtn단추를 찰칵하였을 때의 사건수속*

```
PUBLIC SUB FunBtn_Click()
 ProgressBar1.Value = ProgressBar1.Value + 0.01
 IF ProgressBar1.Value > 0.99 THEN
  ProgressBar1.Value = 0.0
 ENDIF
END
```

- *NoFunBtn단추를 찰칵하였을때의 사건수속*

```
PUBLIC SUB NoFunBtn_Click()
  ProgressBar1.Value = ProgressBar1.Value - 0.01
  IF ProgressBar1.Value < 0.01 THEN
    ProgressBar1.Value = 0.0
  ENDIF
END
```

이와 같이 모든 코드작성이 끝나면 프로젝트를 보관하고 실행단추를 찰칵한다. 그러면 아래와 같이 프로그람이 시작되는것을 보게 될것이다.

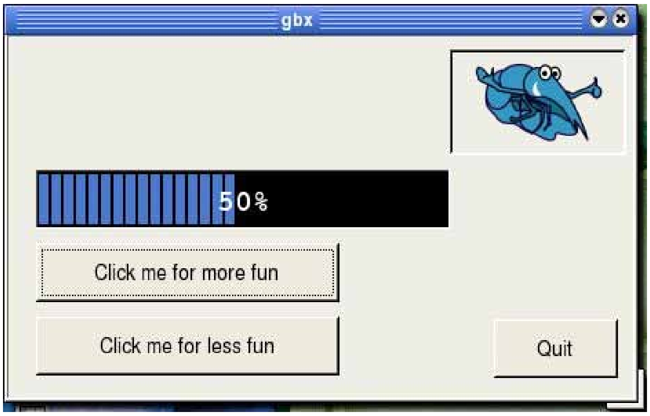


그림 6-10. 실례프로그람

현재 "Click me for more fun" 단추를 찰칵하면 상태띠가 변한다.
그것을 3번 찰칵하면 그림 6-11과 같이 된다.

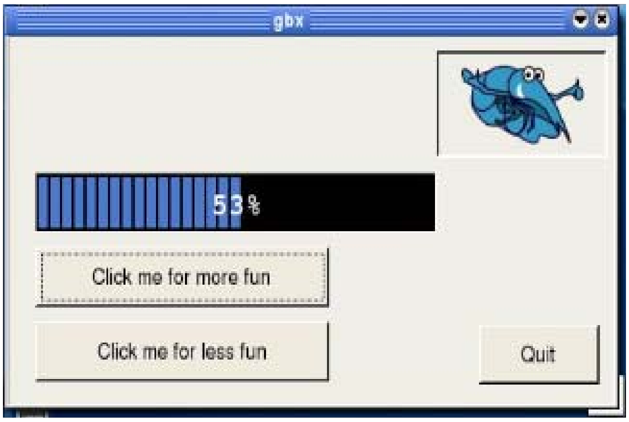


그림 6-11. FunBtn단추를 3번찰칵했을 때의 상태

또한 마우스가 푸른 새우그림우에서 왔다갔다하는데 따라 표식이 생겼다없어졌다하는것을 볼수 있다.

마지막으로 프로그람을 끝내기 위하여 ESC건이나 Quit단추를 찰칵할수 있다.

이 실례에서는 마지막으로 double_click(두번찰칵사건)에 대하여 고찰한다.

- *FunBtn단추를 두번 찰칵했을 때의 사건수속*

```
PUBLIC SUB FunBtn_DblClick()
  ProgressBar1.Value = ProgressBar1.Value + 0.1
  IF ProgressBar1.Value > 0.90 THEN
    ProgressBar1.Value = 0.0
  ENDIF
END
```

- *NoFunBtn단추를 두번 찰칵했을 때의 사건수속*

```
PUBLIC SUB NoFunBtn_DblClick()
 ProgressBar1.Value = ProgressBar1.Value -0.1
 IF ProgressBar1.Value < 0.1 THEN
    ProgressBar1.Value = 0.0
 ENDIF
END
```

현재의 프로젝트를 보관한 다음 실행단추를 찰칵한다.

이때 단추를 찰칵하면 상태띠값이 1씩 증가하지만 두번찰칵하면 상태띠값이 10씩 증가한다.

프로젝트를 보관한 다음 Gambas에서 탈퇴한다.

실례에서 리용한 사건들은 Click, DblClick, Enter, Leave사건들이다.

이 외에도 마우스끌기와 떨구기조종에 리용할수 있는 Drag, DragMove, Drop, LostFocus, GotFocus사건과 건반조종과 관련된 keypress , keyrelease사건들도 있는데 이것은 나중에 더 구체적으로 보기로 한다.

5. 도형사용자대면부(GUI)프로그람작성방법

우의 실례프로그람작성과정을 통하여 도형사용자대면부프로그람작성단계를 다음과 같이 구분할수 있다.

① 문제요구분석 및 프로젝트설계단계(학습장에서)

이 단계에서는 해결하려는 문제의 조건과 요구를 구체적으로 분석하며 프로젝트의 형태를 결정하고 프로젝트를 구성하게 될 홈과 클라스, 모듈구성방안을 설계한다.

② 대면부설계단계(IDE환경에서)

홈의 륜곽을 설계하고 필요한 콘트롤들을 배치한다.

다음 객체들(홈과 콘트롤)의 속성을 규정한다.

③ 코드작성단계(코드창에서)

처리해야 할 사건들을 구분하고 사건수속들을 완성한다.

④ 오유수정단계(IDE환경에서)

프로그람을 실행시키면서 오유를 수정한다.

⑤ 프로젝트의 보관(IDE환경에서)

완성된 프로젝트의 원천을 보관한다.

⑥ 콤파일단계(IDE환경에서)

오유가 없다고 인정되면 프로그람을 콤파일하여 실행파일로 만든다.

앞으로 이 책에서 프로그람작성은 ①~③단계까지를 구분하여 서술한다.

제 7 장. 자료입력을 위한 콘트롤

6장에서는 그림콘트롤, 표식콘트롤, 단추콘트롤, 상태띠콘트롤들과 그의 사건들에 대하여 취급하였다.

이러한 형태의 콘트롤들은 모두 자료를 표시하거나 마우스나 단추를 누를 때 일어나는 사건에 응답하지만 사용자들이 목록에서 어떤 항목을 선택하거나 필요한 자료를 입력하는것과 같은 작업은 할수 없다.

이 장에서는 자료입력에 리용되는 콘트롤들의 대표적인 속성과 메쏘드, 사건을 리용하는 방법에 대하여 서술한다.

그리고 장의 마지막에는 이 콘트롤들을 종합적으로 리용하는 프로그람을 작성하게 된다.

① TextBox (본문칸)콘트롤

한행으로 된 본문자료를 입력하거나 표시하는데 리용되는 콘트롤이다.

이 클라스는 코드로도 창조할수 있다.

```
DIM hTextBox AS TextBox
hTextBox = NEW TextBox(Parent AS Container)
```

maxlength속성

입력되는 문자수의 웃한계를 설정하는 속성으로서 표준값은 0이다. 이것은 본문칸에 입력할수 있는 최대문자수를 의미한다.

Alignment속성

본문칸에서 문자의 맞추기기준을 설정하는 속성이다.

Left:왼쪽맞추기

Right:오른쪽맞추기

Center:중심맞추기

Password속성

이 속성을 True로 설정하면 본문칸에 입력되는 모든 문자들이 별표(*)로 표시된다. 이 속성은 사용자암호와 같은것을 입력할 때 리용할수 있다.

Clear메쏘드

본문칸의 내용을 모두 지운다.

② TextLabel(본문표식)콘트롤

간단한 HTML본문을 표시하는 콘트롤이다.

이 클라스는 코드로도 창조할수 있다.

```
DIM   hTextLabel   AS   TextLabel
hTextLabel = NEW   TextLabel(Parent AS Container)
```

이 콘트롤에서는 HTML언어를 리용하여 본문의 표시형식을 여러가지로 조종할수 있다.

[실례 1]

TextLabel1.Text = "<div align=center> " & TextBox1.Text & ""

TextBox1에 입력된 본문은 TextLabel1에서 HTML형식에 따라 다시 형식화되여 나타나게 된다.

현재는 사용자가 TextBox1에 입력하는 본문이 자동적으로 TextLabel1의 중심에 굵은 글자체로 현시된다.

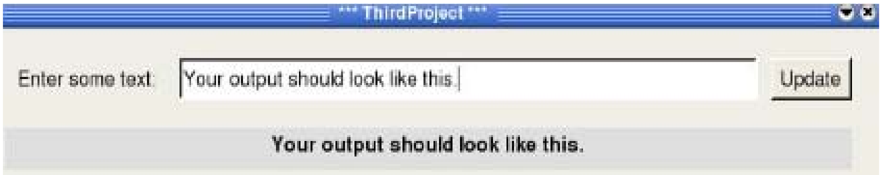


그림 7-1. HTML형식화를 리용하여 수정된 TextLabel결과

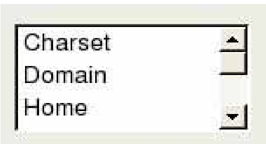


그림 7-2. 실례ListBox

③ ListBox(목록칸)콘트롤

선택할수 있는 여러개의 본문항목들을 표시하는 콘트롤이다.

실례로 그림 7-2와 같은 형태의 ListBox를 만들자.

도구창에서ListBox콘트롤을 선택하여 홈에 배치한 다음 속성창에서 list속성을 찾는다.

이 속성의 오른쪽에 있는 입력칸을 눌러 목록속성편집대화칸을 연다. 목록에 항목을 넣기 위해 편집기의 밑에 있는 본문칸에 항목을 입력한다.

목록에서 첫 항목으로 Charset를 입력한 다음 Insert단추를 찰칵한다.

같은 방법으로 목록에 Domain, Home, Host, Language, Path, User를 추가한다.

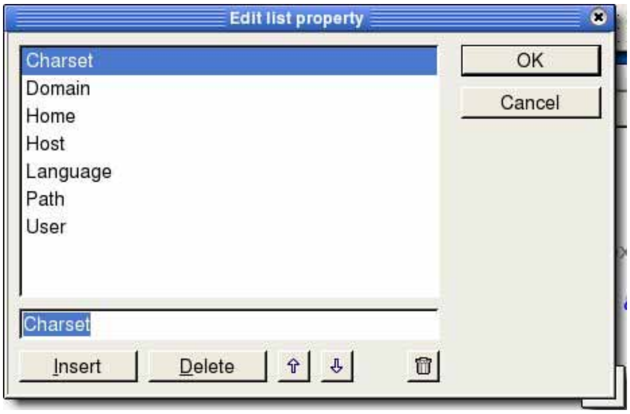


그림 7-3. ListBox의 목록속성편집대화창

그림 7-3과 같이 항목들이 차례로 정돈되여 들어갔는가를 확인한 다음 OK단추를 찰칵한다.

이 클라스는 코드로 창조할수도 있다.

```
DIM hListBox AS ListBox
hListBox = NEW ListBox(Parent AS Container)
```

Count속성

목록에 등록된 항목의 개수를 돌려주는 속성이다. 항목의 개수를 셀 때에는 제일 첫 항목을 0으로 하여 세기 시작한다는것을 항상 명심하여야 한다.

List(첨수)속성

목록칸에 등록되여있는 항목들을 선택할수 있게 해주는 속성이다.

[실례 2]

```
ss= listBox1.list(1)
```

이때 변수 ss에는 목록칸의 두번째 항목의 내용이 들어간다.

Text속성

목록칸에서 선택된 항목을 돌려준다.

Add메쏘드

형식 add(항목 [, 첨수])

목록칸에 첨수로 지적된 위치에 기호렬로 된 항목을 추가하는 메쏘드이다.

첨수는 0부터 시작된다.

만일 첨수가 지적되지 않으면 목록칸의 제일 마지막항목으로 추가된다.

[실례 3]

```
listBox1.Add(“김순희”, 1)
```

Remove메쏘드

형식 Remove(첨수)

첨수로 지적된 항목을 목록칸에서 삭제한다.

Refresh메쏘드

목록칸의 내용을 다시 현시하여주면서 목록칸의 속성들을 재정리한다.

목록칸에서 프로그람적으로 항목들을 추가하거나 제거한 경우에는 refresh메쏘드를 호출한 다음에야 count속성값을 리용할수 있다.

④ ComboBox(복합칸)콘트롤

본문칸과 목록칸이 결합된 콘트롤로서 건반으로부터 문자를 입력할수도 있고 목록으로부터 항목을 선택할수도 있다.

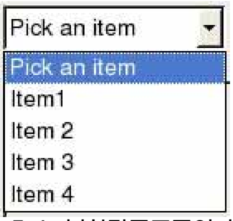


그림 7-4. 복합칸콘트롤의 실례

홈에 복합칸을 만드는것은 목록칸을 만드는것과 비슷하지만 조금 차이나는 점도 있다.

실례로 홈에 그림 7-4와 같은 복합칸을 만들어보자.

도구창에서 ComboBox콘트롤을 선택하여 홈에 배치한 다음 속성창문으로 가서 list속성을 찾는다.

이 속성의 오른쪽에 있는 입력칸을 찰칵하면 목록속성편집을 위한 대화칸이 열린다. 이때 Pick an item이 첫 항목으로 항상 표준적으로 나와있는데 이것을 그대로 리용하려면 Insert단추를 찰칵한다.

목록에 Item1, Item2, Item3, Item4를 추가한다.



그림 7-5. 목록속성편집기

그림 7-5에서 보는바와 같이 항목들이 차례로 정돈되여 들어갔는가를 확인한 다음 OK단추를 찰칵한다.

목록이 다 추가된 다음 홈에서 보면 복합칸은 목록칸과 달리 마치도 본문칸처럼 보인다.



그림 7-6. 설계시의 ComboBox

이 클라스는 코드로 창조할수도 있다.

```
DIM hComboBox AS ComboBox
hComboBox=NEW ComboBox(Parent AS Container )
```

복합칸의 속성과 메쏘드들은 목록칸과 거의 비슷하다.

목록칸이나 복합칸과 같은 선택콘트롤들의 가장 큰 우점은 사용자들이 자료를 부정확하게 타자할수 있는 가능성을 감소시켜준다는것이다.

⑤ Frame(틀)콘트롤

경계를 가지는 용기와 같은 콘트롤이다.

이 클라스는 코드로 창조할수도 있다.

```
DIM  hFrame  AS  Frame
hFrame = NEW Frame(Parent  AS  Container)
```

Frame에서 중요한것은 그것이 창문안의 창문과 같이 동작한다는것이다.

Frame에 고정시킨 임의의 콘트롤은 Frame의 한 부분으로 된다.

즉 Frame안에 단추나 검사칸과 같은 다른 콘트롤을 배치하면 그것들은 Frame이 움직일 때 함께 움직인다.

그림 7-7에 여러개의 콘트롤을 포함한 Frame의 실례를 주었다.



그림 7-7. Frame의 실례

⑥ ToggleButton(토글단추)콘트롤

한번 찰칵하면 설정되고 다시 한번 찰칵하면 설정이 해제되는것과 같은 기능을 수행하는 단추콘트롤이다. 이 콘트롤은 일부 전자제품들에서 하나의 단추로 켜고 다시 한번 눌러 끄는 동작을 모방한것이다.

이 클라스는 코드로 창조할수도 있다.

```
DIM  hToggleButton  AS  ToggleButton
hToggleButton = NEW ToggleButton(Parent AS Container)
```

Value속성

이 속성은 단추의 누름상태를 반영한 론리형속성이다. 만일 ToggleButton단추를 찰칵하면 Value속성값이 TRUE로 설정되고 다시한번 찰칵하면 FALSE로 설정된다.

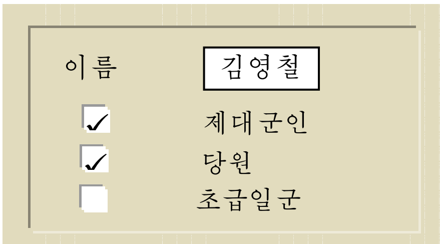


그림 7-8. 검사칸콘트롤의 실례

⑦ Checkbox(검사칸)콘트롤

어떤 대상에 대하여 on/off로 선택하게 하는 콘트롤로서 두가지 경우를 가진 대상들의 상태를 구분하여 표시할 때 리용한다.

실례로 매 학생에 대하여 제대군인인가/아닌가, 당원인가/아닌가, 초급일군인가/아닌가와 같은 내용을 확인하려고 할 때 검사칸콘트롤을 리용할수 있다.

이 클라스는 코드로 창조할수도 있다.

```
DIM  hCheckBox  AS  CheckBox
hCheckBox = NEW CheckBox(Parent  AS  Container)
```

Value속성

검사칸의 상태를 반영한 론리형속성이다. 사실 검사칸을 누를 때마다 Value속성값이 TRUE와 FALSE로 서로 바뀌는데 검사칸에 검사기호가 있을 때에는 Value속성값이 TRUE로 설정된다.

⑧ Panel(판)

여러개의 콘트롤들을 묶어 배치할수 있는 용기와 같은 기능을 수행하는 콘트롤로서 틀과 비슷하지만 표식이 없는 차이점이 있다.

실례로 그림과 같이 단일선택단추들을 묶어 배치할수 있다.

이 클라스는 코드로 창조할수도 있다.

```
DIM hPanel AS Panel
hPanel = NEW Panel (Parent AS Container)
```



그림 7-9. 단일선택단추를 가진 Frame

⑨ RadioButton(단일선택단추)콘트롤

여러가지 선택가능성들가운데서 꼭 하나만을 선택하려고 할 때 리용되는 콘트롤이다. 그러므로 단일선택단추는 대부분 하나가 아니라 여러개를 하나의 그룹으로 묶어 리용한다.

그룹화는 단일선택콘트롤들을 Panel에 배치하는 것으로 실현한다.

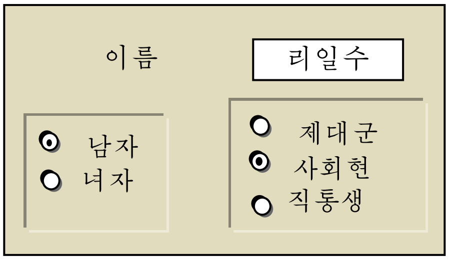


그림 7-10. 단일선택단추의 실례

실례로 학생의 성별은 남자/녀자 중 어느 한 항목에 해당된다.

또한 학생경력도 제대군인/사회현직/직통생중 어느 한 항목에 해당된다.

따라서 성별을 하나의 그룹으로, 학생경력을 하나의 그룹으로 묶어 단일선택단추들을 배치할수 있다.

단일선택단추들을 Panel에 그룹화해놓으면 프로그람실행시에 선택된 단일선택단추를 제외한 나머지 단추들은 자동적으로 선택이 해제된다. 즉 그룹가운데서 꼭 하나만이 선택된다.

이 클라스도 코드로 창조할수 있다.

```
DIM hRadioButton AS RadioButton
hRadioButton=NEW RadioButton(Parent AS Container)
```

프로그람작성실례

① 프로젝트설계

프로젝트이름: ThirdProject

형태: 도형사용자대면부

폼이름: Form1

- 시작클라스
- 모든 콘트롤들을 public로 한다.

대면부설계도

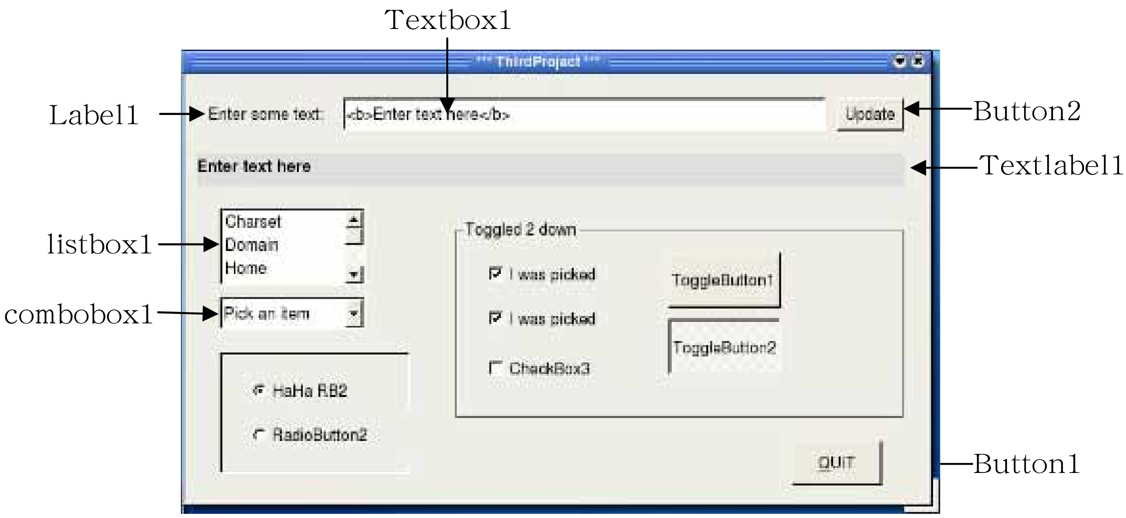


그림 7-11. 세번째프로젝트의 대면부

② 대면부설계

Button1	name	Quitbtn
	Text	QUIT
Button2	name	Updatebtn
	Text	Update
Label1	Text	Enter some text

③ 코드작성

- 홈을 열 때의 사건수속

```
PUBLIC SUB Form_Open()
  ME.Caption = " *** ThirdProject *** "
END
```

- Quit단추를 찰칵할 때의 사건수속

```
PUBLIC SUB QuitBtn_Click()
  Form1.Close
END
```

- Update단추를 찰칵할 때의 사건수속

```
PUBLIC SUB UpdateBtn_Click()
   TextLabel1.Text = "<div align=center> <b>" & TextBox1.Text & "</b>"
END
```

- *마우스를 TextBox1에 가져갈 때의 사건수속*

```
PUBLIC SUB TextBox1_enter() .
   TextBox1.Clear
END
```

- *마우스가 TextBox1을 벗어날 때의 사건수속*

```
PUBLIC SUB TextBox1_Leave() .
    TextLabel1.Text = "<div align=center> <b>" & TextBox1.Text & "</b>"
    TextBox1.Text = "<Enter text here>"
End
```

코드의 첫행은 HTML의 형식을 리용하여 TextBox1의 문자렬을 중심에 배치하고 굵은 서체로 설정하였다.

두번째행에서는 마우스가 본문칸을 떠날 때 <Enter text here>라는 통보가 항상 나타나도록 하였다.

- *마우스로 폼을 한번 찰칵하였을 때의 사건수속*

```
Form_MouseDown()
   UpdateBtn.Hide
End
```

이 코드에 의해 사용자가 단추를 사용할 필요가 없을 때 폼을 한번 눌러 단추를 숨기도록 하였다.

- *마우스로 폼을 두번찰칵했을 때의 사건수속*

```
Form_DblClick()
   UpdateBtn.Show
End
```

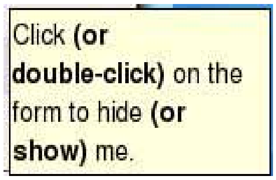


그림 7-12. ToolTip 실례

사용자가 Update단추를 리용하여야 할 경우에는 폼을 두번 찰칵한다.

폼을 한번 찰칵하여 Update단추를 숨기고 두번찰칵하여 다시 보이게 하는 동작은 프로그람을 처음으로 리용하는 사용자는 알수 없다. 이것을 ToolTip속성으로 해결할 수 있다.

Update단추를 선택하고 속성창문을 연 다음 ToolTip속성을 찾는다.

세점(...)이 있는 회색의 오른쪽항목을 선택하면 새로운 편집창문이 펼쳐진다.

여기에 다음과 같은 본문을 입력한다.

Click (or double-click) on the form to hide (or show) me.

마우스를 Update단추에 가져가면 그림 7-12와 같은 형식화된 통보문이 현시된다.

속성창에서 세점이 있는 항목은 새로운 대화칸을 펼칠수 있다는것을 의미한다.

- ComboBox의 항목을 선택할 때의 사건수속

사용자가 ComboBox에서 무엇인가를 선택하면 항목은 변화되며 이 변화는 Change사건을 발생한다. 여기에 아래와 같은 코드를 입력한다.

```
PUBLIC SUB ComboBox1_Change()
   DIM result AS String
   result = "<div align=center> <b>" & Trim(ComboBox1.Text) & "</b>"
   TextLabel1.Text = result & " was selected from the ComboBox"
 END
```



그림 7-13. 추가삭제단추의 추가.

복합칸에 새로운 항목의 추가나 삭제를 위해 PlusBtn와 MinusBtn이라는 두개의 단추를 홈에 추가한다.(그림 7-13)

- PlusBtn단추를 찰칵할 때의 사건수속

```
PUBLIC SUB PlusBtn_Click()
   DIM Item AS String
   DIM NumItems AS Integer
   ComboBox1.Refresh
   NumItems = ComboBox1.Count
   IF NumItems = 0 THEN
       Item = "Pick an Item"
   ELSE
       Item = "Item " & Str(NumItems)
   ENDIF
   ComboBox1.Refresh
   ComboBox1.Add(Item, NumItems)
END
```

- MinusBtn단추를 찰칵할 때의 사건수속

```
PUBLIC SUB MinusBtn_Click()
   DIM NumItems AS Integer
   ComboBox1.Refresh
   NumItems = ComboBox1.Count
   IF NumItems > 0 THEN
      DEC NumItems
      IF NumItems <> 0 THEN
         ComboBox1.Remove(NumItems)
      ENDIF
      ComboBox1.Refresh
   ENDIF
END
```

- ListBox1에서 항목을 선택할 때의 사건수속

```
PUBLIC SUB ListBox1_Click()
```

```
    IF Trim(ListBox1.Text) = "System.Charset" THEN
      TextLabel1.Text = System.Charset
      ELSE IF Trim(ListBox1.Text) = "Domain" THEN
        TextLabel1.Text = System.Domain
      ELSE IF Trim(ListBox1.Text) = "Home" THEN
        TextLabel1.Text = System.Home
      ELSE IF Trim(ListBox1.Text) = "Host" THEN
        TextLabel1.Text = System.Host
      ELSE IF Trim(ListBox1.Text) = "Language" THEN
        TextLabel1.Text = System.Language
      ELSE IF Trim(ListBox1.Text) = "Path" THEN
        TextLabel1.Text = System.Path
      ELSE IF Trim(ListBox1.Text) = "User" THEN
        TextLabel1.Text = System.User
    ENDIF
  END
```

- *ToggleButton1단추를 찰칵할 때의 사건수속*

```
  PUBLIC SUB ToggleButton1_Click()
    IF ToggleButton1.Value = TRUE THEN
      Frame1.Text = "Toggled1 down"
    ELSE
      Frame1.Text = "Toggled1 up"
    ENDIF
  END
```

- *ToggleButton2단추를 찰칵할 때의 사건수속*

```
  PUBLIC SUB ToggleButton2_Click()
   DIM result AS String
    IF ToggleButton2.Value = TRUE THEN
      Frame1.Text = "Toggled2 down"
    ELSE
      Frame1.Text = "Toggled2 up"
    ENDIF
  END
```

코드에서는 ToggleButton단추의 Value속성값을 검사한 다음 그 상태를 Frame1의 본문속성에 나타나게 하였다.

- *CheckBox1을 찰칵할 때의 사건수속*

```
  PUBLIC SUB CheckBox1_Click()
    DIM outline1 AS String
    DIM outline2 AS String
    DIM outline3 AS String

    IF CheckBox1.Value = TRUE THEN
```

```
        Checkbox1.Text  = "I was picked"
    ELSE
        Checkbox1.Text  = "Checkbox1"
    ENDIF
END
```

- *CheckBox2을 찰칵할 때의 사건수속*

```
PUBLIC SUB CheckBox2_Click()
    IF CheckBox2.Value = TRUE THEN
        Checkbox2.Text  = "I was picked"
    ELSE
        Checkbox2.Text  = "Checkbox2"
    ENDIF
END
```

- *CheckBox3을 찰칵할 때의 사건수속*

```
PUBLIC SUB CheckBox3_Click()
    IF CheckBox3.Value = TRUE THEN
        Checkbox3.Text  = "I was picked"
    ELSE
        Checkbox3.Text  = "Checkbox3"
    ENDIF
END
```

- *RadioButton1을 찰칵할 때의 사건수속*

```
PUBLIC SUB RadioButton1_Click()
    RadioButton1.Text = "HaHa RB2"
    RadioButton2.Text = "RadioButton2"
END
```

- *RadioButton2를 찰칵할 때의 사건수속*

```
PUBLIC SUB RadioButton2_Click()
    RadioButton2.Text = "HaHa RB1"
    RadioButton1.Text = "RadioButton1"
END
```

제 8 장. 파일 및 입출력관리

이 장에서는 Gambas에서의 파일관리와 입출력조작에 대하여 취급한다.

파일이란 한마디로 말하여 자료들의 모임이다.

파일관리에는 필요한 파일(등록부)들의 찾기, 파일들의 속성을 알아내기, 새로운 파일(등록부)의 창조, 파일의 갱신, 파일(등록부)제거 등과 같은 내용이 속한다.

여기서 파일의 창조, 갱신과 같은 입출력조작은 파일열기, 자료의 읽기쓰기, 파일닫기라는 세 단계를 반드시 거쳐야 한다.

① 파일열기

디스크에 자료를 쓰거나 디스크로부터 자료를 읽어들이기전에 우선 파일이 열려야 한다.

② 자료의 읽기 및 쓰기

열린 파일로부터 자료를 읽어들이기, 디스크에 자료쓰기, 자료제거, 자료갱신을 한다.

③ 파일의 닫기

파일의 닫기는 파일에 대한 자료처리를 다 진행한 다음 프로그람의 마지막단계에서 진행한다.

1. 입출력관리명령문들

① OPEN

형식

OPEN 파일이름 FOR [READ] [WRITE] [CREATE | APPEND] [DIRECT] [WATCH] AS # 파일지적자

OPEN은 자료의 읽기쓰기 및 창조, 추가를 위하여 파일을 열기한다.

READ: 자료읽기방식으로 파일을 연다.

WRITE: 자료쓰기방식으로 파일을 연다. 이때 파일경로는 도중에 절대로 변하지 말아야 한다.

CREATE: 새로운 파일을 창조하는 방식이다.

지적한 파일이 존재하지 않으면 그 파일이름으로 새로운 파일이 창조된다.

그러나 지적한 파일이 존재하면 그 파일의 내용은 없어지고 같은 이름을 가진 새로운 파일이 창조된다.

APPEND: 이 방식으로 파일을 열면 곧 파일지적자가 파일의 끝으로 이동하며 순차적으로 자료를 추가할수 있다.

DIRECT: 이 방식으로 파일을 열면 읽기쓰기를 다 진행할수 있다.

WATCH: 이 방식으로 열면 다음과 같은 조건을 리용하여 파일을 검사할수 있다.

파일로부터 적어도 한byte라도 읽을수 있으면 File_Read()사건이 호출되며 파일에 적어도 한byte 쓸수 있으면 File_Write()사건이 호출된다.

파일열기를 할 때에는 CREATE방식이 지적되지 않는 한 파일이 반드시 존재하여야 한다.

파일지적자: 이 파라메터는 파일객체를 지적하는 변수이다.

앞으로 파일에 대한 자료의 읽기쓰기명령문을 리용할 때에는 해당한 파일지적자를 리용해야 한다.

② CLOSE

> 형식
>
> CLOSE # 파일지적자

CLOSE는 OPEN과 반대되는 함수로서 열려진 파일을 닫는다.

파일을 닫을 때에는 다음과 같은 뒤처리가 진행된다.

우선 완충기에 남아있는 마지막부분의 자료를 물리적으로 디스크에 쓴다. 다음 자료파일의 끝에 끝표식 EOF를 써넣은 다음 완충기를 해제한다. 다음 파일지적자를 해제한다.

OPEN과 CLOSE는 프로그람에서 항상 짝패처럼 함께 리용된다.

③ INPUT

> 형식
>
> INPUT 변수1 [, 변수2 …]
>
> 또는
>
> INPUT # 파일지적자, 변수1 [,변수2 …]

표준입력으로부터 자료를 읽어 변수들에 차례로 넣는다.

또는 파일지적자로 지적된 파일로부터 자료를 읽어들여 변수들에 차례로 넣는다.

④ LINE INPUT

> 형식
>
> LINE INPUT 변수
>
> 혹은
>
> LINE INPUT # 파일지적자, 변수

표준입력으로부터 변수에 한개의 본문행을 읽어넣는다.

혹은 파일로부터 변수에 한개의 본문행을 읽어넣는다.

[실례 1]

```
STATIC PUBLIC SUB Main()
  DIM hFile AS File
  DIM sInputLine AS String
  DIM lineCtr AS Integer

  OPEN "The Raven.txt" FOR READ AS # hFile
  'Eof()함수는 파일의 끝을 검사하는 함수이다.
 '아래에서 구체적으로 본다.
WHILE NOT Eof(hFile)
    LINE INPUT # hFile, sInputLine
    PRINT Str(lineCtr) & Chr(9) & sInputLine
    INC lineCtr
  WEND
  CLOSE # hFile
END
```

[주의]

2진파일에는 행바꾸기기호가 없으므로 2진파일을 읽을 때에는 LINE INPUT명령문이 아니라 READ명령문을 리용해야 한다.

⑤ READ

형식 READ 변수[, 길이] 혹은 READ # 파일지적자, 변수[, 길이]

READ는 파일 또는 표준입력으로부터 자료를 읽어들이는 함수이다.

READ는 입구자료를 2진자료로 취급한다.

이때 2진자료의 자료형은 읽기조작을 하는 동안에 자료가 보관되는 변수의 자료형에 따라 결정된다.

자료의 표현형식은 WRITE명령문을 사용할 때의 표현형식과 같아야 한다.

만일 변수가 기호렬이면 읽을 byte수를 길이로 지적해야 한다.

또한 길이의 값이 부수이면 자료파일끝에서부터 지적한 byte수만큼 읽어들이며 길이값을 지적하지 않으면 자료파일로부터 전체 기호렬을 읽어들인다.

[실례 2]

```
READ # hFile, OneLine, -256
```

여기서 -256은 읽어들이려고 하는 최대byte수이다.

-256대신에 다른 수를 쓸수 있지만 그 수를 읽은 자료를 기억하기 위한 변수의 최대크기로 제한시켜야 한다.

⑥ PRINT

형식

PRINT 식 1[(; | ,) 식 2 …] [(; | ,)]

혹은

PRINT # 파일지적자, 식 1[(; | ,) 식 2 …] [(; | ,)]

표준출력 혹은 파일지적자로 지적된 파일에 식의 값을 출력한다.

이때 《;》은 련이어 기록한다는것을 의미하며 《,》은 Tab길이만큼 공백을 두고 출력한다는것을 의미한다.

⑦ WRITE

형식

WRITE 식[, 길이]

혹은

WRITE # 파일지적자, 식[, 길이]

WRITE는 표준출력 또는 파일에 대한 쓰기를 진행한다.

만일 출구하려는 식이 문자렬인 경우 쓰기해야 할 byte수를 길이로 지적해야 한다.

[실례 3]

```
STATIC PUBLIC SUB Main()
dim I as integer
dim st as string
dim file1,file2,f1 as file

OPEN system.Path &/ "infile.txt" FOR WRITE AS #file1
FOR i = 1 TO 10
  WRITE #file1, Str(i)
NEXT
CLOSE #file1

OPEN "/usr/infile.txt" FOR READ AS #file2
```

```
  FOR i = 1 TO 10
  INPUT #file2, st
     ListBox1.Add(st)
  NEXT
CLOSE #file2

OPEN system.Path &/ "/data" FOR CREATE AS #f1
  FOR i = 1 TO 9
    PRINT #f1, Str(i)
  NEXT
CLOSE #f1

OPEN system.Path &/ "/data" FOR APPEND AS #f1
  FOR i = 10 TO 20
    PRINT #f1, Str(i)
  NEXT
CLOSE #f1
End
```

⑧ SEEK

형식 SEEK #파일지적자, 위치

SEEK명령문은 다음번 읽기/쓰기조작을 위한 자료지적자의 위치를 정의하여준다.

만일 위치에 지적한 값이 부수이면 자료지적자는 파일끝에서부터 지적한 위치만큼 거꾸로 움직인다.

지적자를 파일의 끝으로 움직이려면 Lof()함수를 리용한다.

[실례 4]

```
'hFile파일의 시작위치에로 이동
SEEK #hFile, 0
'hFile파일의 마지막 다음위치에로 이동
SEEK #hFile, Lof(#hFile)

'hFile파일끝으로부터 100byte위치에로 이동
```

```
SEEK #hFile, -100
```

⑨ FLUSH

> 형식
> FLUSH [#파일지적자]

FLUSH명령문은 파일의 자료완충기내용을 완료하기 위하여 호출한다.

파일지적자를 지적하지 않으면 모든 파일들에 대하여 완충기내용을 다 완료한다.

대체로 FLUSH는 쓰기조작을 여러번 진행한 후에 프로그람이나 부분프로그람을 닫기전에 완충기에 기억되여있던 모든 자료들을 모두 파일에 쓰기할 목적으로 리용한다.

2. 파일관리를 위한 표준함수들과 명령문들

파일관리에 필요한 표준함수들과 명령문들을 소개한다.

① Access

> 형식
> Access(경로[, 방식])

Access는 파일에 접근할수 있는가를 결정하는 함수이다.

경로로 지적한 파일에 접근할수 있으면 이 함수는 TRUE값을 돌려준다. 방식으로는 gb.Read, gb.Write, gb.Exec를 지적할수 있다.

방식이 gb.Read로 설정되였을 때 파일을 읽을수 있으면 함수는 TRUE값을 돌려준다.

방식이 gb.Write로 설정되였을 때 파일에 쓰기할수 있으면 함수는 TRUE값을 돌려준다.

방식이 gb.Exec로 설정되였을 때 파일이 실행될수 있으면 함수는 TRUE값을 돌려준다.

등록부에 대하여 gb.Exec를 지적하면 등록부를 열람할수 있다는것을 의미한다.

또한 우의 방식들을 OR연산자로 조합하여 쓸수 있다.

[실례 5]

```
PRINT Access("/home/rabbit", gb.Write OR gb.Exec)
PRINT Access("/root", gb.Write)
```

② Dir

> 형식
> Dir(등록부경로 [, 파일패턴])

등록부경로에 있는 파일들가운데서 파일패턴에 맞는 파일들의 이름을 문자렬묶음으로 돌려주는 함수이다.

파일패턴이 지적되지 않은 경우에는 등록부에 있는 모든 파일들의 이름을 돌려준다.

파일패턴에서는 LIKE연산자를 리용하여 다음의 기호들을 조합하여 쓸수 있다.

기호	의 미
*	임의의 형태의 임의의 개수의 글자들
?	임의의 형태의 꼭 한개의 글자
[abc]	괄호안에 지적된 임의의 글자
[x-y]	괄호안에 지적한 구간에 놓이는 임의의 글자
[^x-y]	괄호안에 지적하지 않은 구간에 놓이는 임의의 글자

이외에 기호《\》는 그뒤에 오는 문자를 패턴기호로 해석하지 못하게 한다.

[실례 6]

```
'등록부내용인쇄
SUB PrintDirectory(Directory AS String)
  DIM sFile AS String

  FOR EACH sFile IN Dir(Directory, "*.*")
    PRINT sFile
  NEXT
END
```

③ Eof

형식

  Eof(파일지적자)

파일의 끝에 도달한 경우 론리값 TRUE를 돌려주는 함수이다.

[실례 7]

```
OPEN FileName FOR READ AS #hFile
WHILE NOT Eof(hFile)
  LINE INPUT #hFile, OneLine
  PRINT OneLine
WEND
```

```
CLOSE #hFile
...
```

④ Exist

형식 Exist(경로)

파일이나 등록부가 존재하면 TRUE값을 돌려주는 함수이다.
만일 지적한 경로가 존재하지 않으면 FALSE를 돌려준다.

[실례 8]

```
...
   DIM sCurrentFileName AS String

   sCurrentFileName = "Raven.txt"
   IF Exist(sCurrentFileName) THEN
... '파일열기
   ELSE
     Message.Info("파일- " & sCurrentFileName & " 이 존재하지 않음.")
   ENDIF
...
```

⑤ IsDir/Dir?

형식 IsDir(경로)

경로에서 지적한것이 등록부인 경우에는 TRUE 값을 돌려주는 함수이다. 경로가 존재하지 않거나 등록부가 아니면 함수는 FALSE값을 돌려준다

[실례 9]

```
PRINT IsDir("/etc/password")
   False

PRINT IsDir(Application.Home &/ ".gambas")
   True

PRINT IsDir("/windows")
   False
```

⑥ Stat

형식
Stat(경로)

파일이나 등록부에 대한 정보를 돌려주는 함수이다.

파일에 대한 정보로서는 형태, 크기, 마지막수정날자 등이 있다.

이 함수에 다음과 같은 파일 속성들이 리용될수 있다.

.Group .Hidden .LastAccess .LastChange .LastUpdate
.Mode .Perm .SetGID .SetUID .Size
.Sticky .Time .Type .User

[실례 10]

```
WITH Stat("/home")
   PRINT .Type = gb.Directory
   PRINT Round(.Size / 1024); "K"
END WITH
```

결과

```
True
4K
```

⑦ Temp/Temp$

형식
Temp$()

림시적인 파일에 붙일수 있는 유일한 이름을 돌려준다. 만들어진 파일이름은 /tmp 등록부에 존재한다.

[실례 11]

```
PRINT Temp$()
```

결과

```
/tmp/gambas.0/12555.1.tmp
```

⑧ Lof

형식
Lof(파일지적자)

열려진 파일의 길이를 돌려준다.

[실례 12]

```
'Gambas Class File sCurrentFileName AS String sCurrentFileLength AS Integer

PUBLIC SUB Form_Open()
 DIM sInLine AS String
 DIM hFileIn AS File

sCurrentFileName = "The Raven.txt"
IF Exist(sCurrentFileName) THEN
  OPEN sCurrentFileName FOR READ AS hFileIn
   sCurrentFileLength = Lof(hFileIn)
   WHILE NOT Eof(hFileIn)
      LINE INPUT #hFileIn, sInLine
      TextArea1.Text = TextArea1.Text & Chr(13) & Chr(10) & sInLine
   WEND
  CLOSE hFileIn
ELSE
  Message.Info("File " & sCurrentFileName & " does not exist.")
ENDIF
END
```

⑨ COPY

형식
COPY 원천파일경로 TO 목적파일경로

COPY함수는 원천파일을 목적파일로 복사한다.

이때 목적파일경로가 원천파일경로와 일치하면 안된다.

이 함수를 리용하여 등록부는 복사할수 없다.

[실례 13]

```
'환경설정파일을 보관한다
COPY System.Home &/ ".gambas/gambas.conf" TO "/mnt/save/gambas.conf.save"
```

⑩ KILL

```
형식
    KILL 경로
```

KILL함수는 경로에 존재하는 파일을 완전히 제거한다.

⑪ RENAME

```
형식
    RENAME 본래파일이름 AS 새파일이름
```

RENAME은 파일이나 등록부의 이름을 새 이름으로 변경한다.

⑫ MKDIR, RMDIR

```
형식
    MKDIR 등록부이름
    RMDIR 등록부이름
```

MKDIR함수는 파일체계에서 등록부를 창조할 때 리용하며 RMDIR는 등록부를 제거할때 리용된다.

[실례 14]

```
MKDIR "file:/rittingj/My Documents/Gambas/test"
RMDIR "file:/rittingj/My Documents/Gambas/test"
```

제 9 장. 도형처리

Gambas에서는 DrawingArea콘트롤과 PictureBox콘트롤, MovieBox콘트롤을 리용하여 정지화상 혹은 동화상을 실현한다.

여기서 PictureBox콘트롤은 단순히 이미 주어진 화상파일을 표시하는 기능을 수행한다. MovieBox콘트롤도 이미 주어진 동화상파일을 재생하는 기능을 수행한다.

그러나 DrawingArea콘트롤에서는 우의 두 콘트롤과는 달리 이미 주어진 그림이 아니라 프로그람작성자의 요구대로 그림을 그리거나 도형처리를 할수 있다.

DrawingArea콘트롤에서 도형처리는 Draw클라스, picture클라스, image클라스를 리용하여 진행한다.

미술가가 그림을 그리는것과 비교하여 말하면 DrawingArea콘트롤은 그림을 그리는 화판이고 picture클라스와 image클라스는 빈 그림종이 또는 그림을 그린 종이장이라고 말할수 있으며 Draw클라스는 붓, 수채화구와 같은 그림그리기도구라고 말할수 있다.

미술가가 그림을 그리기전에 화판과 그림도구들을 준비하는것처럼 해당 DrawingArea콘트롤에 그림을 그리려면 그리기전에 Draw클라스의 Begin메쏘드를 먼저 호출해야 한다.

또한 도형처리를 끝내려면 Draw클라스의 End메쏘드를 반드시 호출해야 한다.

이 장에서는 도형처리에 리용되는 콘트롤과 클라스에 대하여 그의 대표적인 속성과 메쏘드를 구체적으로 학습하고 프로그람작성과정을 통하여 정통하도록 한다.

1. DrawingArea 콘트롤

DrawingArea는 도형처리를 할수 있는 콘트롤로서 그리기구역을 나타낸다.

속성으로는 Name, X, Y, Width, Height, Visible, Enablel, Font, Background, Foreground, Mouse, Border, Cached 가 있으며 메쏘드로서는 Clear, Delete, Drag, Grab, Hide, Lower, Move, Raise, Refresh, Resize, Setfocus, Show가 있다.

대부분 속성들과 메쏘드들은 앞에서 다른 콘트롤들을 학습할 때 취급한것이므로 여기서는 일부 대표적인것만 설명한다.

① Background, Foreground속성

이 속성들은 DrawingArea콘트롤의 배경색과 전경색을 설정하는 속성이다.

② Cached속성

그리기구역의 내용이 기억기에 고속완충된다는것을 지적하여주는 론리형속성이다.

이 속성을 True로 설정하면 VB에서 AutoRedraw속성을 리용할 때와 같이 그림그리는 결과가 즉시 반영되며 또한 다른 창문에 의해 가리워졌던 부위도 인차 재생된다.

도형을 지우려고 clear메쏘드를 리용하는데 이때에도 Cached속성을 True로 설정해주는것이 좋다.

[실례 1]

Drawingarea1.Cached = TRUE

③ clear메쏘드

Clear메쏘드는 DrawingArea콘트롤의 그림을 지우며 모든 화소점들을 령으로 설정한다.

④ refresh메쏘드

refresh메쏘드는 DrawingArea콘트롤에 그림을 현시하여준다.

[실례 2]

```
Drawingarea2.clear
…
Drawingarea2.refresh
```

⑤ Resize메쏘드

형식
Resize(너비, 높이)

DrawingArea콘트롤의 크기를 설정한다.

2. Draw 클라스

1) 속성

① FillColor

FillColor는 다각형을 그릴 때 다각형의 내부색을 설정하거나 돌려주는 옹근수형속성이다.

② FillStyle

다각형의 내부를 색칠할 때 그 내부형태를 설정하거나 돌려주는 옹근수형속성이다.

0~14까지의 옹근수를 줄수 있다.

내부형태를 표현하는데 리용되는 표준적인 상수들도 있다.

BackDiagonal	Cross	CrossDiagonal	Dense12
Dense37	Dense50	Dense6	Dense63

Dense88 Dense94 Diagonal Horizontal

None Solid Vertical

③ FillX와 FillY

FillX와 FillY속성들은 다각형의 내부를 그릴 때 시작점의 X/Y좌표를 설정하거나 돌려주는 옹근수형속성이다.

④ Font

Font는 본문표시에 리용되는 서체를 설정하거나 돌려준다.

[실례 3]

```
Draw.Font.Name = "Lucida"
Draw.Font.Bold = TRUE
Draw.Font.Italic = FALSE
Draw.Font.Size = "10"
Draw.Font.StrikeOut = FALSE
Draw.Font.Underline = FALSE
```

⑤ Invert

Invert속성은 그리려는 도형들의 화소색을 반전하겠는가를 규정하는 론리형속성이다.

⑥ LineStyle와 LineWidth

LineStyle속성은 도형의 테두리선을 그릴 때 선의 형태를 설정하거나 돌려주는 옹근수형속성이다. 0~5까지의 옹근수값을 설정할수 있다.

LineStyle속성에서 리용할수 있는 상수들도 정의되여있다.

Dash Dot DashDot

DashDotDot None Solid

LineWidth속성은 선의 굵기를 설정하거나 돌려주는 옹근수형속성이다.

수값이 클수록 선의 굵기가 커진다.

⑦ Transparent(투명도)

Transparent속성은 그리기구역에 본문을 표시할 때 배경색이 나타나도록 하겠는가 아니면 나타나지 않도록 하겠는가(즉 투명하게 보이도록 하겠는가)를 지적하는 론리형속성이다.

2) 메쏘드

① begin

형식
draw.begin(D)

D로 지적한 그리기구역에 대한 그림그리기시작을 선포한다.

여기서 D는 DrawingArea콘트롤을 지적하는 객체형변수를 의미한다.

② end

```
형식
    draw.end
```

그리기구역에 대한 그림그리기를 끝낸다.

③ text

```
형식
    draw.text(문자렬, x, y[, 너비, 높이, 위치맞추기])
```

그리기구역의 x, y위치에 파라메터에서 지적한 너비, 높이를 가진 직4각형테두리안에 문자렬을 표시하고 위치맞추기(left, right, center)를 진행한다.

④ Point

```
형식
    draw.point(x, y)
```

draw.point는 그리기구역의 x, y위치에 하나의 화소점을 그린다.

⑤ rect

```
형식
    draw.rect(x, y, 너비, 높이)
```

draw.rect는 x, y를 시작점으로 하고 파라메터에서 지적한 너비, 높이를 가지는 직4각형을 그린다.

⑥ line

```
형식
    draw.line(X1, Y1, X2, Y2)
```

Line은 직선을 그린다.

x1, y1는 선분의 시작위치 x2, y2는 선분의 끝위치를 나타낸다.

⑦ ellipse

```
형식
    Ellipse (X, Y, 너비, 높이[, 시작각, 부채형의 각])
```

이 메쏘드는 타원과 원, 부채형을 그린다.

원은 타원에서 너비와 높이가 같은 경우로 본다.

x, y: 원 또는 타원의 중심위치

너비: 타원의 수평직경(너비)

높이: 타원의 수직직경(높이)

시작각: 부채형을 그릴 때 부채형의 시작각.

부채형의 각: 부채형을 그릴 때 부채형의 각 .

⑧ Polygon

> 형식
>
> draw.Polygon(점자리표 AS Integer[ ])

Polygon메쏘드는 n개의 정점을 가진 다각형을 그린다.

점자리표는 정점들의 자리표를 포함하는 옹근수묶음이다.

자리표들은 x, y형식으로 되여있으므로 다각형의 매 정점에 대하여 두개의 옹근수가 대응되게 된다.

⑨ Polyline

> 형식
>
> draw.Polyline(정점자리표 AS Integer[])

draw.Polyline는 n개의 정점으로 이루어진 꺾인선을 그린다.
N개의 정점에 첫 정점을 다시 첨가하면 다각형이 그려진다.

⑩ tile

> 형식
>
> draw.tile(그림 AS Picture, x, y, 너비, 높이)

Tile메쏘드는 마치도 벽에 타일을 붙이듯이 그리기구역의 x, y위치에 지적한 너비와 높이로 그림을 《붙인》다.

⑪ image

> 형식
>
> draw.image(화상 AS image, x, y)

draw.image메쏘드는 화상을 그리기구역의 지적한 위치에서부터 그린다.

메쏘드를 습득하는 가장 좋은 방법은 매개 메쏘드들을 리용하는 코드를 작성하여보는것이다.

이를 위하여 Draw클라스에 있는 거의 모든 메쏘드들의 사용방법을 보여주는 실례프로그람을 작성하여보자.

[실습 1]

① **프로젝트설계**

프로젝트 이름 : gfxDemo

형태: 도형사용자대면부

홈이름: Form1

- 시작클라스
- 모든 콘트롤들을 public로 한다.

대면부설계도



그림 9-1. gfxDemo홈의 설계도

② 대면부설계

Form1	width	530
	height	490
Button1	name	TextBtn
	text	Text
	width	60
	height	25
Button2	name	InvRectBtn
	text	InvRect
	width	60
	height	25
Button3	name	EllipseBtn
	text	Ellipses
	width	60
	height	25
Button4	name	FillRectBtn
	text	FillRect
	width	60
	height	25
Button5	name	Btn
	text	FillPoly
	width	60
	height	25
Button6	name	PolyLineBtn
	text	PolyLine
	width	60
	height	25
Button7	name	TileBtn

```
             text       Tileimg
             width      60
             height     25
Button8      name       QuitBtn
             text       Quit
             width      60
             height     25
DrawingArea  name           da
```

대면부설계도를 참고하면서 우의 속성을 설정한다.

③ 코드작성

- *constructor사건수속*

```
PUBLIC SUB _new()
 '그리기구역의 너비, 높이설정
  da.W = form1.W - 10
  da.H = form1.H - 45
 '그리기 시작
  Draw.Begin(da)
 '선두께를 체계설정으로 2로 설정
  Draw.LineWidth = 2
END
```

Constructor는 객체가 초기화될 때 실행되는 프로그람이다.

따라서 프로그람이 기동하여 홈이 열리기전에 _new()사건수속이 먼저 호출되며 그다음 Form_Open사건수속이 실행된다.

_new()사건수속에서는 대체로 자료의 초기화, 프로그람의 초기화공정이 진행된다.

- *홈열기 사건수속*

```
PUBLIC SUB Form_Open()
   Form1.Text = " Drawing Examples "
END
```

- *Quit단추를 찰칵할 때의 사건수속*

```
PUBLIC SUB QuitBtn_Click()
   Draw.End '그리기 끝내기
   ME.Close ' Form1닫기
END
```

- *Text단추를 찰칵할 때의 사건수속*

```
PUBLIC SUB TextButton_Click()
   DIM counter AS Integer              '순환계수변수
   da.Clear                            '현재 그리기구역 지우기
   Draw.ForeColor = color.Black        '전경색을 검은색으로 설정
```

```
    draw.Font.Name = "Arial"           '현재의 서체를 Arial로 설정
    draw.Font.Bold = TRUE              '굵은체로 설정
    draw.Font.Italic = FALSE           '사선체를 해제
    draw.Font.Underline = FALSE        '밑선체를 해제
    draw.Font.Size = "16"              '글자크기를 16으로 설정
    FOR counter = 0 TO 9               '순환을 시작하여 10번 순환
                                 '본문을 출구하고 y위치를 40씩 증가
      draw.Text("Sample Text", 10, 10 + 40*counter, 100,60)
                              '본문의 너비와 높이는 각각 100, 60로 설정
    NEXT                               '순환끝
    draw.Font.Size = "24"              '글자크기는 24로 설정
    Draw.ForeColor = color.Blue     '전경색을 푸른색으로 설정
    FOR counter = 0 TO 9               '두번째 순환도 10번
           '첫번째 순환과 차이나는것은 본문의 x위치를 200으로 설정
      draw.Text("More Sample Text", 200,10 + 40*counter, 100,60)
    NEXT
    da.Refresh                      '그리기구역 연시
  END '
```

Text단추를 찰칵하였을 때의 실행결과

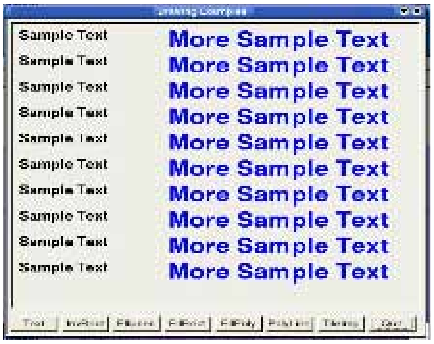


그림 9-2. Text 단추를 찰칵했을 때의 결과

- *InvRect단추를 찰칵할 때의 사건수속*

```
    PUBLIC SUB InvRectBtn_Click()
      DIM i AS Integer

      da.Clear                          '그리기구역 지우기
              'fill style을 None으로 설정. 즉 그릴때 아무런 형태도 없다.
      draw.FillStyle = fill.None

      draw.ForeColor = color.cyan    '그리기색을 청록색으로 설정
      draw.Invert = TRUE                '청록색사각형이 붉은색으로 나타난다.
      FOR i = 0 TO 15 '16번순환
```

```
                    '직4각형의 시작점위치와 끝점위치를 5씩 증가한다.
      Draw.Rect(5+5*i, 5+5*i, da.W/2+5*i, da.H/2 + 5*i)
                        '짝수번째순환에서는 Invert속성을 바꾸어준다.
      IF i MOD 2 = 0 THEN
        draw.Invert = FALSE
      ELSE
        draw.Invert = TRUE
      ENDIF
    NEXT 순환끝

    draw.Invert = FALSE              'invert을 off로 설정 (색반전 중지)
    draw.Forecolor = color.Black     '현재 전경색을 검은색으로 설정
    draw.Point(da.W/2, da.H/2)       '화면의 가운데 점 그리기
  '방금 그린 첫 중심점으로부터 한 화소수만큼 상하좌우로 움직여 십자선을 그린다.
    draw.Point(da.W/2-1, da.H/2)
    draw.Point(da.W/2+1, da.H/2)
    draw.Point(da.W/2, da.H/2-1)
    draw.Point(da.W/2, da.H/2+1)
    da.Refresh                       '그림 연시.
  END
```

실행결과

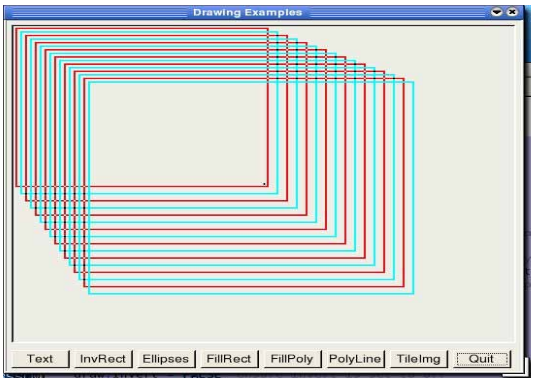


그림 9-3. InvRect 단추를 찰칵하였을 때의 결과
화면중심에 작은 검은 십자점

- *Ellipses 단추를 찰칵할 때의 사건수속*

```
PUBLIC SUB EllipseBtn_Click()
  DIM x1 AS Integer                '작업변수 소개
  DIM x2 AS Integer
  DIM y1 AS Integer
  DIM y2 AS Integer
  DIM i AS Integer                 '순환변수 소개
              '선의 두께를 변화시킬 때 선두께값을 보관하기 위한 변수
  DIM lwidth AS Integer
```

```
    da.Clear                          '시작전에 그리기구역 지우기
    x1 = 50                           '초기값설정
    y1 = 50
    x2 = 100
    y2 = 100
    Draw.ForeColor = color.DarkBlue   '전경색을 검푸른색으로 설정
    draw.FillStyle = fill.Solid       '채우기 형태를 solid방식으로 설정
    FOR i = 0 TO 360 STEP 30          '12번까지의 순환을 설정
      IF i MOD 45 = 0 THEN            '각이 90도의 배수이면 색설정을 변경
        draw.FillColor = color.Yellow
      ELSE
        draw.FillColor = color.Red
      ENDIF
            ' I값으로 시작하여 30도 회전한 12개의 타원토막을 그린다.
      Draw.Ellipse(x1, y1, x2, y2, i, 30)
    NEXT
    draw.FillStyle = fill.None'fill style을 없앤다.
    draw.ForeColor = color.Black        '그리기전경색을 검은색으로 설정
             '몇개의 선분들을 그리자, x자리표=화면의 수평중심- 25
    x1 = da.W/2-25
    y1 = da.H/2                       'y자리표= 화면의 수직중심위치
    lwidth = 1                        '선분의 두께는 1픽셀로 설정
    FOR i = 10 TO 200 STEP 25
                           'y축을 따라 25픽셀씩 움직이면서 8번 순환
      draw.Line(da.W/2, i, da.W/2 + 150, i)     '선분그리기
      INC lwidth                        '선분의 두께변수 증가
      draw.LineWidth = lwidth           '새로운 두께 설정
    NEXT                                '순환끝
    draw.LineWidth = 2                '표준그리기 두께를 2로 설정
                           '다른 순환을 리용하여 몇개의 타원을 그리자
    FOR i = 1 TO 40 STEP 3 '13개 만든다
      Draw.Ellipse(x1-i*5, y1, 75, 200)
                              '수평축을 따라 한번 왼쪽으로 이동
      Draw.Ellipse(x1+i*5, y1, 75, 200)
                              '수평축을 따라 한번 오른쪽으로 이동
      IF i MOD 2 = 0 THEN             '순환에서 짝수번째이면 색갈변화
        draw.ForeColor = color.Red
      ELSE
        draw.ForeColor = color.Black
      ENDIF
    NEXT                                '순환끝
da.Refresh                            '그리기한 내용표시
END                              '끝
```

- *Ellipses 단추를 찰칵하였을때의 실행결과*

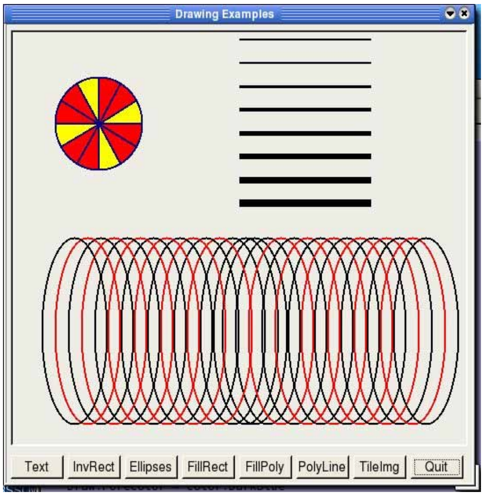


그림 9-4. Ellipses 단추를 찰칵하였을 때의 결과

- *FillRect 단추를 찰칵할 때의 사건수속*

```
PUBLIC SUB FillRectBtn_Click()
   DIM x1 AS Integer                  '국부변수 소개
   DIM x2 AS Integer
   DIM y1 AS Integer
   DIM y2 AS Integer
   da.Clear                 '그리기구역 지우기

   x1 = 20                  '직사각형의 왼쪽웃구석점의 자리표설정
   y1 = 80
   x2 = x1+80               '직사각형의 오른쪽아래구석점의 자리표설정
   y2 = y1+150
   draw.FillColor = color.Red        '그림그리기색을 붉은색으로 설정
   draw.FillStyle = fill.Horizontal  '수평선형채우기형태 설정
'직4각형을 그린다. 이때 수평선형태로 문양이 채워진다.
   Draw.Rect(x1,y1,x2, y2)
   x1 = 220                  '직사각형의 왼쪽꼭대기구석점 설정
   y1 = 180
   x2 = x1+300              '직사각형의 오른쪽아래구석점 설정
  y2 = y1+300
  draw.FillColor = color.Blue          '그리기색을 푸른색으로 설정
  draw.FillStyle = fill.Cross          '십자형형태로 채우기형태설정
'직4각형을 그린다 . 이때 십자선형태의 문양이 채워진다.
```

```
 draw.Rect(x1,y1,x2,  y2)
  draw.FillStyle = fill.None                    '채우기 형태를 없앤다
  da.Refresh                                  '작업과정을 화면으로 보여준다
END
```

실행결과

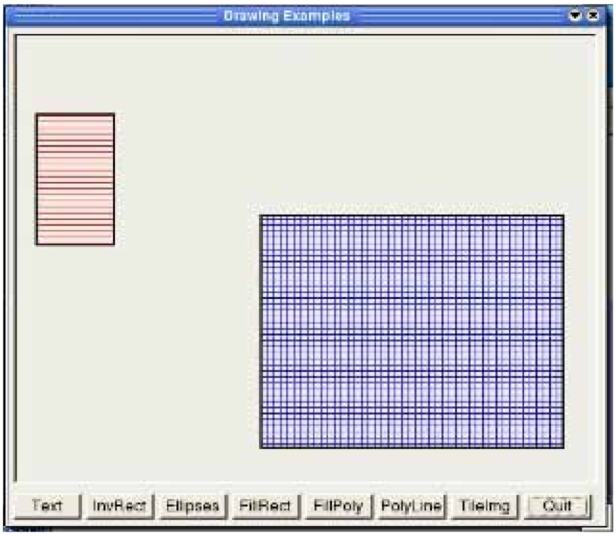


그림 9-5. FillRect 단추를 찰칵하였을 때의 결과

- *FillPoly 단추를 찰칵할 때의 사건수속*

```
PUBLIC SUB PolygonBtn_Click()
   DIM x1 AS Integer
   DIM y1 AS Integer
   DIM x2 AS Integer
   DIM y2 AS Integer
   DIM x3 AS Integer
   DIM y3 AS Integer

  ' 3각형의 정점들을 옹근수묶음변수triangle에 보관하려고 한다.
   DIM triangle AS Integer[]
   da.Clear
                              '3각형의 매 정점들을 설정
      '2등변3각형의 (첫정점)꼭두점 설정
   x1 = da.w/2 'x1:도형그리기구역의 수평중심
   y1 = 10 'y1:그리기구역의 꼭대기에서 10아래위치
      '2등변3각형의 왼쪽정점(두번째점)설정
    x2 = da.Left + 10
   y2 = da.H - 200
       '2등변3각형의 오른쪽정점(세번째점) 설정
   x3 = da.W - 20
```

```
    y3 = y2
                  '정의된 모든 정점들이 묶음변수에 기억된다.
    triangle = Array(x1, y1, x2, y2, x3, y3)
    draw.FillStyle = fill.Dense63               '채우기형태설정
    draw.FillColor = color.DarkMagenta            '채우기 색갈설정
                       'triangle묶음자료를 리용하여 3각형을 그린다.
   draw.Polygon(triangle)
   draw.FillStyle = fill.None                   '채우기형태를 없앤다.
   da.Refresh                                '그리기 한 내용을 현시
 END                                         '끝
```

실행결과

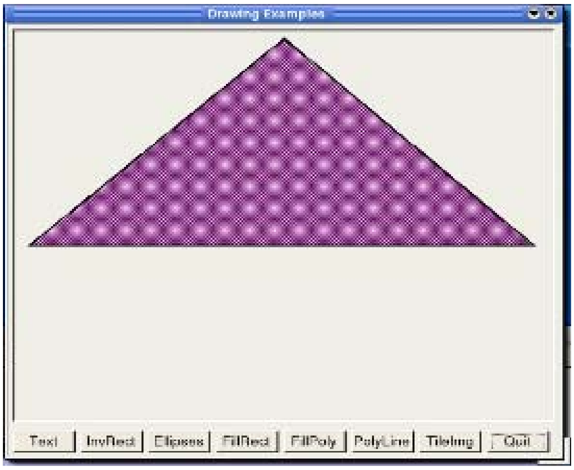


그림 9-6. Draw.Polygon을 리용한 3각형그리기

- *Polyline 단추를 찰칵할 때의 사건수속*

```
 PUBLIC SUB PolyLineBtn_Click()
    DIM x1 AS Integer
    DIM y1 AS Integer
    DIM x2 AS Integer
    DIM y2 AS Integer
    DIM x3 AS Integer
    DIM y3 AS Integer
    DIM x4 AS Integer
    DIM y4 AS Integer
    DIM lines AS Integer[]       '꺾인선자료를 위한 옹근수묶음변수소개

    da.Clear                     '그리기구역 지우기
    x1 = da.w/2                  '첫정점
    y1 = 10
    x2 = da.Left + 10            '다음정점
    y2 = da.H    200
```

```
    x3 = da.W - 20                '3번째정점
    y3 = da.H - 20
    x4 = da.W/2                   '4번째정점은 그리기구역의 중심
    y4 = da.H/2
                                  '정점들을 묶음에 보관.
   '만일 다각형을 만들려면 첫정점 x1, y1을 묶음에 더 추가
                                  '즉 다음행의 설명문을 해제
    lines = Array(x1, y1, x2, y2, x3, y3, x4, y4, x1, y1)
    lines = Array(x1, y1, x2, y2, x3, y3, x4, y4)
    draw.FillStyle = fill.None    '채우기형태 없애기
    draw.ForeColor = color.DarkGreen   '채우기색을 검은풀색으로 설정
    draw.Polyline(lines)          '선분그리기
    da.Refresh                    '그림표시
END
```

실행결과

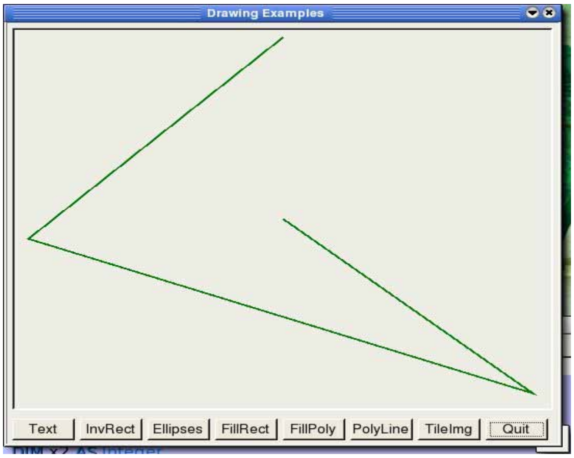


그림 9-7. Draw.Polyline의 사용실례

3. Picture 클라스

Picture클라스는 그림을 표시하는데 리용되는 클라스이다.
이 클라스는 코드로 창조할수도 있다.

[**실례** 4]

```
DIM hPicture AS Picture
hPicture = NEW Picture([너비, 높이, 투명성])
```

만일 너비와 높이가 지적되지 않으면 새로운 picture는 비게 된다.
투명성파라메터에는 그림을 투명하게 하겠는가에 따라 론리값을 지적할수 있다.

1) Picture클라스의 속성

Picture클라스의 속성들로는 Depth, Height, Width, Image 등이 있다.

① Depth

그림의 농도속성을 돌려주거나 설정해준다.

농도값을 설정할 때 2, 8, 16, 24값만을 설정할수 있다.

② Height

읽기만 하는 속성으로서 그림의 높이를 옹근수값으로 돌려준다.

③ Width

읽기만 하는 속성으로서 그림의 너비를 돌려준다.

④ Image

읽기만 하는 속성으로서 그림을 image로 변환하고 그것을 돌려준다

2) Picture클라스의 메쏘드

Picture클라스의 메쏘드로서는 Clear, Copy, Fill, Flush, Load, Resize, Save 등을 들수 있다.

① Clear

그림을 지우고 모든 화소점들을 령으로 한다.

② Copy

형식

```
Copy([X, Y, 너비, 높이]) AS Image
```

그림 혹은 그 부분을 복사하여 돌려준다.

③ Fill

형식

```
Fill(색)
```

지적된 색갈로 그림구역을 채운다.

④ Load와 Save

형식

```
Load(파일경로)
Save(파일경로)
```

파일로부터 그림을 적재하거나 그림을 파일에 보관한다.

파일경로에서 파일확장자는 보관되는 파일의 형식을 지적해준다.

현재 가능한 파일형식으로서는 JPEG, PNG, BMP, GIF, XPM 등이다.

⑤ Resize

형식
Resize(너비, 높이)

파라메터에서 지적한 너비와 높이에 따라 그림의 크기를 다시 정한다.

[실습 2]

gfxDemo프로그람으로 가서 마지막 단추의 코드작성을 끝내여보자.

gfxDemo프로젝트의 Form1에서 TileImg단추를 두번 찰칵하여 코드창을 열고 다음의 코드를 삽입한다.

Tileimg단추를 찰칵할 때의 사건수속

```
PUBLIC SUB TileBtn_Click()
                                   '국부변수 소개
                    '적재하는 그림을 보관하기 위한 picture변수 소개
  DIM mypic AS Picture
  DIM counter AS Integer              '순환작업을 위한 계수기변수
  DIM i AS Integer
  DIM j AS Integer
  i = 0
  j = 0
     'picture변수를 실체화하면서 너비,높이를 설정하여 비지 않게 한다.
  mypic = NEW Picture(170, 105,FALSE)
  da.Clear                               '그리기구역 지우기
  mypic.Load("Gambas Shrimp.png") 'load메쏘드를 리용하여 그림적재
                         '4행에 3렬로 된 타일그림을 만들려고 한다 .
                         '그러므로 12번의 순환이 필요하다.
  FOR counter = 0 TO 11
    draw.Tile(mypic, i*175, j, 170,105)          '타일형그리기
    INC I                                        '계수기값증가
        '순환계수기값이 3의 배수인가 즉 새행시작인가를 검사한다.
    IF i MOD 3 = 0 THEN
              '계수기값이 3의 배수이면 y축 값 j을 더 증가시킨다.
      j = j + 110
      i = 0                              '렬계수기값을 다시 령으로
    ENDIF
  NEXT '순환끝
  da.Refresh                             '그리기구역에 현시
END                                   'TileImg 단추누르기사건끝
```

실행결과

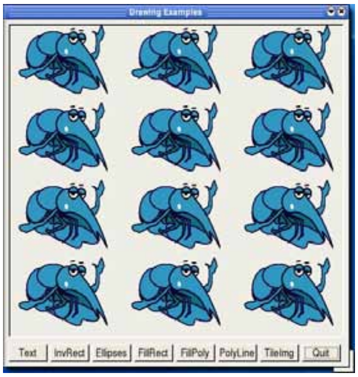


그림 9-8. TileImg 단추를 눌렀을 때 타일형의 그림표시

여기서 중요한것은 프로그람에서 사용하게 될 화상을 선정하는것이다.

이를 위해 푸른새우그림을 찾아 선택하고 그것을 gfxDemo 프로젝트에 "Gambas shrimp.png"라는 이름으로 복사한다.

4. Image 클라스

Image클라스는 화상을 표시하는데 리용되는 클라스이다.

Image클라스와 picture클라스는 비슷하면서도 일정한 차이가 있다.

Picture형태의 그림은 변화시킬수 없지만 Image형태로 된 화상은 확대, 축소, 회전과 같은 변환을 할수 있다.

그러므로 두 클라스는 다 같이 그림을 표시하지만 그 처리방식에서 차이가 있으며 이것은 메쏘드리용에서 나타난다.

Image클라스에는 Depth, Height, Width와 같은 속성들이 있는데 이것들은 picture클라스의 속성과 비슷하다.

Image클라스에는 또한 Clear, Copy, Fill, Flip, Load, Mirror, Replace, Resize, Rotate, Save, Stretch와 같은 메쏘드들도 있다. 여기서는 우와 반복되지 않는 메쏘드들만 구체적으로 설명한다.

① Flip

형식
Flip() AS Image

화상을 수평축에 관하여 대칭한 모습을 돌려준다.

② Mirror

형식

    Mirror() AS Image

화상을 거울복사하여 돌려준다. 즉 수직축을 기준으로 하여 대칭한 모습을 돌려준다.

③ Replace

형식

    Replace(현재색, 새로운색)

현재의 색값을 지적한 새로운 색으로 교체한다.

④ Rotate

형식

    Rotate(각도 AS Float) AS Image

각도만큼 회전한 화상을 돌려준다. 각도는 라디안각이 아니다.

⑤ Stretch

형식

    Stretch(너비, 높이[, 윤활성]) AS Image

너비, 높이만큼 확장한 화상을 돌려준다. 윤활성파라메터에 True를 지적하면 느슨하게 확장된 화상이 얻어진다.

[실습 3]

탁상화면을 보관하였다가 보여주는 프로그람을 작성하여보자.

Gambas를 기동하고 도형사용자대면부방식으로 gfxDemo2라는 이름을 가진 새로운 프로젝트를 창조한다.

IDE환경에서 Form1을 시작홈으로 선정하고 Form1의 resize속성을 TRUE로 하여 화상을 펼쳐볼수 있게 한다.

DrawingArea콘트롤을 홈의 왼쪽웃구석에 배치하고 너비와 높이를 1inch정도로 설정한다. 홈의 오른쪽아래구석에는 단추를 배치하고 《표시》라는 표제를 달아준다.

이 단추를 찰칵하였을 때의 사건수속

```
' Gambas class file
PUBLIC SUB Button1_Click()
 p AS NEW Picture                        'picture변수 선언
 i AS NEW Image                     '화상변수 선언
         'Grab메쏘드를 사용하여 탁상화면으로부터 그림객체를 얻는다
 p = Desktop.Grab()
         '얻은 그림을 image속성을 리용하여 화상형태로 기억시킨다.
```

```
    i = p.image
            '그리기구역의 내용이 기억기에 고속완충되여 있다는것을 지적
    Drawingarea1.Cached = TRUE
            '변화된 화상의 크기만큼 그리기구역의 크기를 다시 설정
    DrawingArea1.Resize(i.Width, i.Height)
    DrawingArea1.Clear()                    '그리기구역지우기
            '그리기를 시작하기 위하여 Draw.Begin 메쏘드를 호출
    Draw.Begin(DrawingArea1)
            '그리기구역의 원점 0,0로 부터 화상을 그려넣는다
    Draw.Image(i, 0, 0)
            'Draw.End 메쏘드를 리용하여 화상그리기를 끝낸다
    Draw.End
    DrawingArea1.Visible = TRUE             '화상을 보이게 한다
    DrawingArea1.Refresh                    '그리기구역을 현시하여준다
  END
```

5. 자리표계리용

DrawingArea콘트롤에 각종 함수의 그라프를 그리자면 가상자리표계를 도입해야 한다.

VB에서는 Scale메쏘드를 리용하여 홈이나 picture콘트롤에 가상자리표계를 도입하고 그라프처리를 진행한다.

그러나 Gambas에는 이러한 메쏘드가 없다.

그러므로 프로그람으로 자리표계를 도입하기 위한 자리표변환부분을 작성하여 리용해야 한다.

DrawingArea콘트롤의 자리표계는 다음의 그림과 같다.

W: DrawingArea콘트롤의 너비, H: DrawingArea콘트롤의 높이

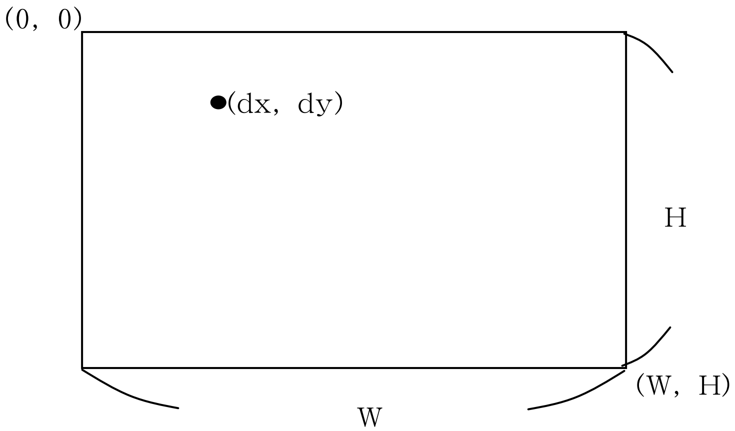


그림 9-9. 그리기구역의 절대자리표계

다음 그림은 같은 DrawingArea콘트롤에 가상자리표계를 도입한것이다.

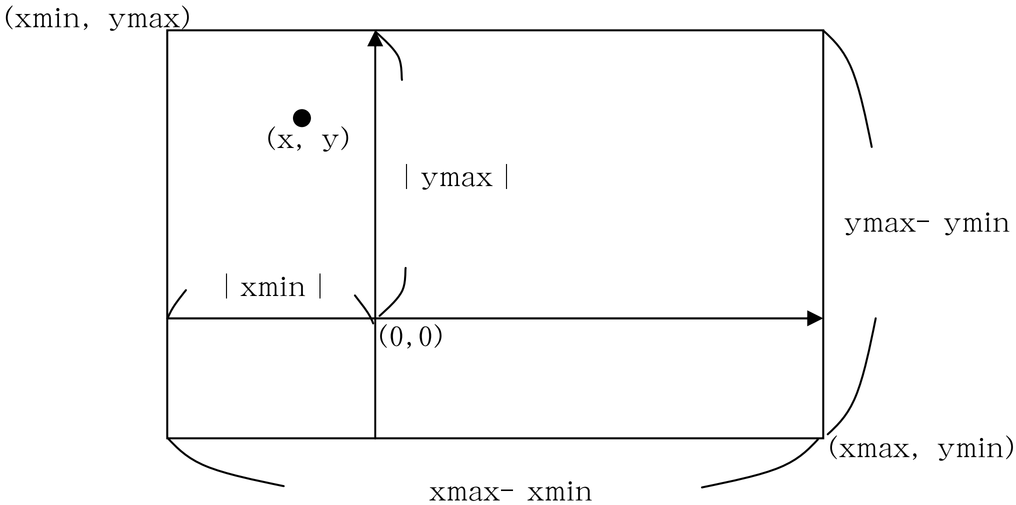


그림 9-10. 그리기구역에 도입된 가상자리표계

가상자리표계의 점(x, y)를 표시하자면 이 점의 자리표를 그리기구역의 자리표(dx, dy)으로 변환해야 한다.

변환공식은 다음과 같다.

$$dx = \frac{W}{xmax-xmin}(|xmin| + x)$$

$$dy = \frac{H}{ymax-ymin}(|ymax| - y)$$

아래의 실례는 VB에서 Scale을 리용하는것과 동등한 처리방법을 보여주고있다.

[실례 5]

y=x^2의 그라프그리기

```
' Gambas class file
public const xmin as integer =-3
public const xmax as integer =3
public const ymin as integer =10
public const ymin as integer =-1

PUBLIC SUB Form_dbclick()
  Me.close
End
```

```
PUBLIC SUB DrawingArea1_dbclick()
  dim x, y AS Float
  dim dx, dy AS Float
  dim H, W as integer
  H = DrawingArea1.Height
  W = DrawingArea1.Width
  Draw.Begin(DrawingArea1)
  For x=xmin to xmax step 0.01
    Y=x*x
    dx=fix(W/(xmax-xmin)*(abs(xmin)+x))
    dy=fix(H/(ymax-ymin)*(abs(ymax)+y))
    Draw.Point(dx, dy)
  next
  Draw.End
end
```

이런 방법으로 가상자리표를 그리기구역의 자리표로 변환하여 요구하는 그라프와 도형에 대한 처리를 진행할수 있다.

제 10 장. Gambas와 자료기지

이 장에서는 Gambas프로그람작성환경에서 자료기지리용방법에 대하여 설명한다.

자료기지를 리용하자면 먼저 관계형자료기지의 기초개념들을 잘 알아야 한다.

자료기지란 어떤 특정의 목적을 위하여 리용하는 자료들을 그의 련관관계를 반영하여 보존한 파일이다.

자료기지관리체계란 이러한 자료기지에 자료를 보관하고 관리(검색, 수정, 삭제)하기 위한 프로그람체계를 말한다.

자료기지는 망형, 계층형, 관계형, 객체형으로 분류한다.

여기서 관계형자료기지는 자료를 표형식으로 표현한다.

실례로 대학교무행정자료기지라고 할 때 그것은 학생명단, 교원명단, 학과명단, 교실배치 등과 같은 여러개의 표들로 구성된다.

매 표는 또한 행과 렬로 이루어진 자료모임구조를 이루고있다.

이때 행은 레코드를 의미하며 렬은 마당을 의미한다.

또한 마당들가운데서 매개 레코드를 정확히 구분할수 있는 마당을 색인마당으로 정한다.

Gambas환경에서는 MySQL, PostgreSQL, SQLite와 같은 관계형자료기지를 리용할수 있다. MySQL은 이 3가지 자료기지들중에서 가장 광범히 리용되는 자료기지이므로 이 장에서는 주로 MySQL을 리용하여 자료기지리용방법을 설명한다.

MySQL은 MySQL server와 client프로그람으로 구성되여있다.

MySQL server는 사용자가 리용하는 모든 자료기지들을 MySQL환경에서 보관하고 관리하는 프로그람이다.

MySQL에 포함된 client프로그람들은 너무 많아서 여기서 렬거하기는 힘들지만 매개 client프로그람들은 모두 MySQL server에 보관된 자료 혹은 메타자료들을 관리할 특별한 목적을 가진 프로그람들이다.

Gambas환경에서 자료기지에 접근하자면 프로젝트에서 gb.db콤포넨트를 리용한다.

그러므로 새로운 프로젝트를 창조할 때 프로젝트속성대화창의 콤포넨트항목에서 gb.db를 선택해야 한다.

gb.db는 PostgreSQL, MySQL, SQLite와 같은 자료기지관리체계들에 접근할수 있게 해준다. PostgreSQL과 MySql들은 다 client/server형프로그람들이므로 이것을 리용한다는것은 곧 자료기지봉사기에 련결할 필요가 있다는것을 의미한다. 그러나 SQLite자료기지인 경우에는 프로그람에서 봉사기에 련결할 필요가 전혀 없다.

gb.db에서 제공하는 클라스들은 Connection, DB, Database, Field, Index, Result, ResultField, Table, User 등이다.

이 클라스들에 대하여 구체적으로 본다.

1. Connection(련결)클라스

이 클라스는 자료기지와의 련결을 보장해준다.

자료기지와 성과적으로 련결하자면 먼저 련결객체를 창조하고 필요한 속성들을 설정한 다음 Open메쏘드를 호출하여야 한다.

이 클라스는 코드로 창조할수도 있으며 형식은 다음과 같다.

```
DIM hConnection AS Connection
hConnection = NEW Connection()
```

1) Connection 클라스의 속성

속성들로는 Charset, Databases, Host, Login, Password, Name, Port, Tables, Type, Users, Version 등이 있다.

① Charset

Charset는 읽기만 할수 있는 속성으로서 자료기지가 리용하는 charset(문자코드모임)를 돌려주는 기호렬속성이다.

② Databases

Databases속성은 자료기지봉사기에 의하여 관리되는 자료기지의 이름들을 포함한 collection객체를 검사하는데 리용한다.

```
PROPERTY Databases AS .ConnectionDatabases
```

③ Host

자료기지봉사기가 자리잡게 될 주콤퓨터이름을 설정하거나 얻기 위하여 이 속성을 리용한다.

주콤퓨터이름은 콤퓨터이름 혹은 IP주소가 될수 있다.

표준적으로 주콤퓨터는 localhost로 되여있다.

이 속성의 자료형은 문자렬형이다.

④ Login

이 속성은 자료기지를 리용하게 되는 사용자의 접속자료를 설정하거나 얻기 위하여 리용한다. 이 속성의 자료형은 문자렬형이다.

⑤ Name

이 속성은 접속하려고 하는 자료기지의 이름을 설정하거나 얻는데 리용한다. 이 속성의 자료형은 문자렬형이다.

⑥ Password

이 속성은 자료기지에 접속할 때 사용되는 암호를 설정하거나 돌려준다. 속성의 자료형은 문자렬형이다.

⑦ Port

이 속성은 자료기지와 련결할 때 사용되는 TCP/IP포구를 설정하거나 돌려주는데 리용한다.

이 속성의 자료형은 문자렬형이다.

⑧ Tables

이 속성은 자료기지에 존재하는 모든 표들의 집합을 얻는데 리용된다.

```
PROPERTY Tables AS .ConnectionTables
```

⑨ Type

이 속성은 련결하려는 자료기지체계의 형태를 표현한다.

Gambas에서 현재 리용할수 있는 자료기지체계는 postgresql, mysql, sqlite이다.

이 속성의 자료형은 문자렬형이다.

⑩ Users

이 속성은 자료기지봉사기에 등록된 모든 사용자들의 집합을 돌려준다.

다른 집합객체들과 마찬가지로 FOR EACH명령문으로 매 사용자를 선택할수 있다.

```
PROPERTY Users AS .ConnectionUsers
```

⑪ Version

이 속성은 련결된 자료기지의 판번호를 돌려주는 읽기만 할수 있는 속성이다.

이 속성의 자료형은 옹근수형이다.

2) Connection클라스의 메쏘드

Connection클라스에는 자료기지와 련결하며 자료의 검색, 추가, 변경, 삭제를 위한 많은 메쏘드들이 준비되여있다.

① Open

형식
자료기지객체.Open() AS Boolean

Open메쏘드는 자료기지와의 련결에 리용한다.

성공적으로 련결된 경우 true값을 돌려준다.

자료기지를 열기에 앞서 자료기지를 리용하는데 필요한 자료와 함께 련결속성을 설정하여야 한다.

일반적으로 최소한 자료기지의 형태, 주콤퓨터, 가입ID암호, 사용하려는 자료기지이름을 설정하여야 한다.

실례 1은 jrittinghouse라는 사용자가 localhost에 보관되여있는 GambasBugs라는 MySQL자료기지와 련결하기 위하여 어떻게 Open메쏘드를 리용하는가를 보여준다.

② close

```
형식
    자료기지객체.close
```

련결된 자료기지와의 해제를 진행할 때 리용된다.

[실례 1]

```
DIM $hConn As NEW Connection
WITH $hConn
    .Type = "mysql"
    .Host = "localhost"
    .Login = "jrittinghouse"
    .Password = "ab32e44"
    .Name = "GambasBugs"
END WITH
TRY $hConn.Open
IF Error THEN PRINT "자료기지를 열수 없습니다. Error =";Error.Text
$hConn.Close
```

만일 자료기지를 련속적으로 리용하고싶다면 자료기지를 다시 열수도 있다.

③ Find

```
형식
    Find(표[, Request AS String, Arguments AS , …]) AS
```

Find메쏘드는 표에서 질문에 해당하여 찾은 결과객체(result object)를 돌려준다.

결과객체는 Gambas가 SQL질문에 대한 결과를 돌려주기 위한 객체이다.

결과객체에 대하여서는 아래에서 더 구체적으로 보도록 하자.

Find메쏘드가 돌려주는 결과객체는 읽기전용이다.

Request: 질문으로 사용되는 SQL WHERE문이다.

Argument: 필요한만큼 라렬하며 Request문자렬안에서 지적된 파라메터들과 교체된다.

즉 Subst()함수의 교체기호렬처럼 작용한다.

Find메쏘드를 사용하면 기초하고있는 자료기지에 관계없이 질문들을 작성할수 있다.

즉 PostgreSQL나 MySQL과 작업한다고 하여도 코드는 달라지지 않는다.

④ Create

```
형식
    Create( 표 AS String ) AS Result
```

Create메쏘드는 표를 창조한다.

이때 창조된 표에서는 레코드들을 창조할수 있으며 읽기/쓰기를 할수 있다.

⑤ Edit

> 형식
>
> Edit(표[, Query AS String, Arguments AS , …]) AS

Edit메쏘드는 표에서 일부 레코드들을 편집하기 위한 읽기/쓰기형 결과객체를 돌려준다.

Query는 표를 려과하기 위하여 리용되는 SQL WHERE문의 조건부와 같다.

Arguments들은 이전에 설명한것과 같다.

[실례 2]

```
DIM MyResult AS Result
DIM sQuery AS String
DIM iParm AS Integer

sQuery = "id=&1"       '질문문자렬의 끝에 파라메터값으로 12를 넣는다.
   iParm = 12

$hConn.Begin                'transaction시작
'표 tblDEFAULT에서 id마당값이 12인 레코드을 찾아 수정하려 한다.
MyResult=$hConn.Edit("tblDEFAULT", sQuery, iParm)
MyResult!Name = "Mr Rittinghouse"        '마당값넣기
$hConn.Update                            '새값으로 수정
$hConn.Commit                            'transaction끝
$hConn.Close                             '자료기지접속끝
```

실례에서 본 업무(Transaction)에 대하여 간단히 설명하겠다.

어느 자료기지에서나 때때로 하나의 조직화된 자료기지처리명령문들이 성과적으로 실행되였는가, 혹은 전혀 실행되지 않았는가를 확인해야 하는 경우가 있다.

은행업무에서 바로 그러한 실례를 찾을수 있다.

만일 주어진 량의 금액(실례로 100원)이 한 구좌에서 다른 구좌로 송금되였다면 다음과 같은 동작이 일어나야 한다.

```
UPDATE account1 SET balance = balance - 100;
UPDATE account2 SET balance = balance + 100;
```

이 두 명령문은 완전히 성과적으로 수행되거나 그렇지 않으면 아예 실행되지 말아야 한다. 왜냐하면 금액은 한 구좌에서만 전송될수 없으며 다른 구좌에로의 송금이 실패하지 말아야 하기때문이다.

이와 같이 반드시 하나로 조직화되여 수행되거나 혹은 실패하는 경우 실행되지 말아야 할 자료기지명령문(질문)들의 묶음을 업무(transaction)라고 한다.

transaction은 BEGIN과 COMMIT사이에 하나 혹은 그 이상의 자료기지명령문들을 묶는것으로 정의한다.

COMMIT명령문이 없이 transaction은 수행되지 않는다.

transaction이 성공적으로 실행되지 못한 경우 rollback명령문을 리용하여 원래상태로 복귀된다.

⑥ Exec

형식
Exec(Request AS String, Arguments AS , …) AS Result

Exec는 SQL로 서술된 질문(request)을 실행시키고 그 결과를 돌려준다.

이때 결과객체는 읽기만 가능한 객체이다.

Request는 SQL WHERE의 조건부와 같다.

⑦ Quote

형식
Quote(Name AS String) AS String

Quote는 SQL질문에 삽입하여 리용할수 있는 식별자들을 돌려준다.

만일 자료기지와 독립적인 코드를 사용하고싶다면 이 메쏘드를 리용할수 있다.

[실례 3]

```
rResult=$hConn.Exec("SELECT COUNT(*) AS Recs FROM " & DB.Quote(DBTable))
PRINT rResult!Recs
```

Dbtable에는 표의 이름이 기억되여있다.

이 실례에서는 지적된 표의 레코드수를 돌려준다.

2. 기타 클라스들

1) Database클라스

자료기지클라스는 자료기지와 관련된 정보를 표현하는데 리용된다.

이 클라스는 창조할수 있다.

이 클라스에서 리용할수 있는 속성들은 Connection, Name, System이다.

Connection은 웃준위의 Connection객체를 얻는데 리용한다.

Name은 자료기지의 이름을 돌려준다.

System은 만일 자료기지가 자료기지체계이면 TRUE를 돌려준다.

자료기지클라스는 자료기지를 제거하는 단 하나의 메쏘드 Delete를 가진다.

2) DB클라스

이 클라스는 현재 접속된 자료기지를 표현한다.

이 클라스는 창조할수 없다.

이 클라스의 대부분속성들은 Connection클라스의 속성들과 거의 같으며 다른 속성들인 Current, Databases, Debug에 대해서만 설명한다.

Current는 현재의 접속을 설정하거나 돌려준다.

Databases는 자료기지봉사기에 의하여 관리되는 모든 자료기지들의 집합을 돌려준다.

Debug는 자료기지콤포넨트가 오유추적방식에 있으면 TRUE를 돌려주거나 설정한다.

Debug기발이 설정되면 임의의 자료기지에서 보내온 매 질문은 조작탁으로 표준오유를 출력한다.

DB클라스에서 리용되는 메쏘드들은 Connection클라스의 메쏘드들과 같다.

3) User클라스

이 클라스는 자료기지의 리용자들을 표현할 때 리용된다.

이 클라스는 창조할수 없다.

이 클라스에는 Administrator, Connection, Name, Password속성들이 있다.

Administrator는 사용자가 자료기지관리자이면 TRUE 값을 돌려주는 읽기전용속성이다.

Connection은 사용자의 웃준위접속객체를 돌려준다.

Name은 사용자의 이름을 돌려준다.

Password는 사용자와 련관된 암호를 설정하거나 돌려준다.

이 클라스에는 자료기지에서 사용자를 제거하는 Delete라는 메쏘드만이 있다.

4) Table클라스

표클라스는 자료기지의 표를 정의한다.

이 클라스는 창조할수 없다.

Connection, Fields, Indexes, Name, PrimaryKey, System, Type 등의 속성이 있다.

Connection은 웃준위의 Connection객체를 얻는데 리용된다.

Fields는 표의 마당들의 집합을 돌려준다.

Indexes는 표의 색인들의 집합을 돌려준다.

Name은 표의 이름을 돌려준다.

PrimaryKey는 기본색인마당의 이름을 포함하는 문자렬묶음을 돌려준다.

System은 자료기지가 자료기지체계이면 TRUE값을 돌려준다.

Type는 표의 형태를 설정하거나 돌려준다.

이 클라스는 단 하나의 메쏘드 Update만을 가진다.

이 메쏘드는 현재 접속된 자료기지에서 표를 실지로 창조할 때 호출된다.

5) Field클라스

마당클라스는 표마당의 자료를 표현하는데 리용된다.

이 클라스는 창조할수 없다.

Default, Length, Name, Table, Type 등의 속성이 있다.

Default는 마당의 표준값을 돌려준다.

Length는 본문마당의 최대길이를 돌려준다.

만일 선택한 본문마당의 길이에 아무런 제한도 지적한것이 없으면 령을 돌려준다.

Name은 마당이름을 돌려주며 Table은 마당이 창조된 표의 이름을 돌려준다.

Type는 마당의 자료형을 지적하는 상수들을 돌려준다.

gb.Boolean, gb.Integer, gb.Float, gb.Date, gb.String

6) Index클라스

Index클라스는 표에 대한 색인을 표현한다.

이 클라스는 창조할수 없다.

이 클라스의 속성들로는 Fields, Name, Primary, Table, Unique 등이 있다.

Fields속성은 첨수화된 마당들을 반점으로 구분한 문자렬을 돌려준다.

Name속성은 표에서 기본색인마당의 이름을 돌려준다.

Table은 이 색인을 가진 표객체를 돌려준다.

Unique는 색인이 유일하면 TRUE값을 돌려준다.

7) Result(결과객체)클라스

결과객체는 SQL질문의 결과를 돌려줄 때 사용되는 클라스이다.

이 클라스는 창조할수 없으며 읽기만 하는 묶음처럼 동작한다.

이 클라스의 매개 객체(레코드)는 FOR EACH문으로 선택하여 리용할수 있다.

결과객체는 다음과 같이 선언하고 리용할수 있다.

```
DIM hResult AS Result
FOR EACH hResult
 '자료를 가지고 무엇이나 한다.
NEXT
```

결과객체에서 사용할수 있는 속성들은 Connection, Count, Fields, Index, Length 이다.

Connection속성

웃준위접속객체를 돌려준다.

Count속성

결과객체의 레코드수를 돌려준다.

Fields속성

결과객체의 마당집합을 돌려준다.

Index속성

령부터 시작된 현재 레코드의 번호를 돌려준다.

Length속성

Count와 꼭같이 리용된다.

결과객체에서 리용하는 메쏘드들은 Delete, MoveFirst, MoveLast, MoveNext, MovePrevious, MoveTo, Update 등이다.

이 메쏘드들중의 대부분은 메쏘드이름만으로도 자체로 리해할수 있으므로 구체적으로 설명하지 않는다.

다만 Update는 자료기지의 현재 레코드를 지적된 값으로 변경할 때 리용하는 메쏘드이다.

또한 MoveTo를 리용할 때 틀린 레코드번호를 지적하면 오유를 의미하는 TRUE 값을 돌려준다는것을 명심하시오.

[**실례** 4]

이 실례는 결과객체의 현재레코드에서 마당값을 얻는 방법을 보여준다.

```
DIM MyResult AS Result
DIM MyVariant AS Variant
MyVariant = MyResult [마당이름 AS String]
```

[실례 5]

이 실레는 결과객체의 현재레코드에서 마당의 값을 변경하는 방법을 보여준다.

```
DIM MyResult AS Result
DIM MyVariant AS Variant
MyResult [마당이름 AS String] = MyVariant
```

8) Resultfield(결과마당)클라스

결과마당클라스는 결과객체의 마당들중 하나를 표현한다.

이 클라스의 속성들로는 Length, Name, Result, Type 등이다.

결과마당자료들을 FOR EACH로 구분할수 있다.

이 방법으로 특별한 마당을 선택하여 Find메쏘드를 리용할수 있다.

이 클라스는 창조할수 없다.

3. 자료기지실레프로그람

여기서는 실레프로그람에 대한 설명을 통하여 Gambas환경에서 자료기지관리프로그람작성방법을 련습하도록 한다.

Gambas를 기동하고 Examples안내에서 Database program을 선택하시오.

IDE환경이 나타나면 이 프로그람이 두개의 홈 FMain과 Frequest를 가지고있다는것을 알수 있다.

Fmain홈에서는 사용하려는 자료기지종류(즉 PostgreSQL, MySQL, SQLite), 주콤퓨터이름, 자료기지이름, 사용자이름, 암호를 입력한다.

홈아래에 있는 SQL질문창이 자료기지에대한 질문을 실행할수 있게 한다.

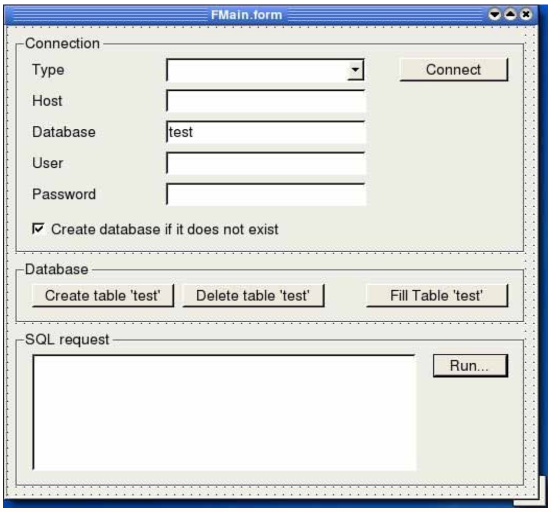


그림 10-1. 설계방식의 FMain홈

FRequest홈은 아래와 류사하다.

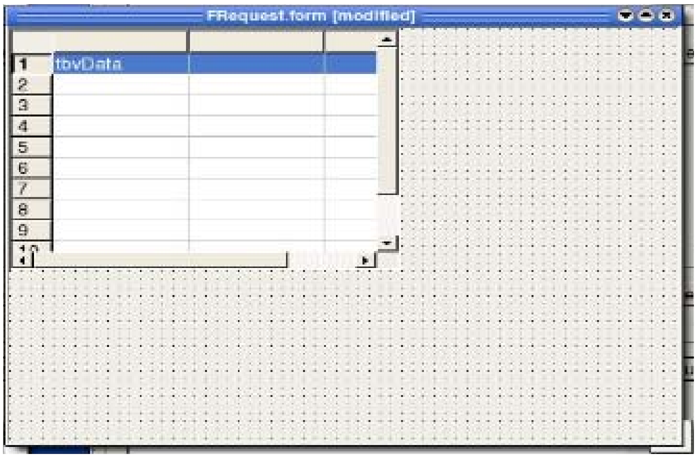


그림 10-2. 설계방식의 Frequest홈

FRequest홈에는 한개의 콘트롤 TableView만이 존재한다.

1) Fmain홈에서의 코드작성

```
' Gambas class file
PRIVATE $hConn AS Connection
```

$hConn는 private변수로 소개되였는데 이 클라스파일의 모든 부분프로그람(함수, 수속)에서 사용되게 된다.

$가 붙어있는 변수는 클라스변수라는것을 명심하시오.

- *connect단추를 찰칵하였을 때의 사건수속*

사용자가 홈에서 Type, Host, Database, User, password항목에 자료를 입력하고 connect단추를 누르면 프로그람은 다음의 사건수속에 와서 우의 자료를 가지고 자료기지에 접속하려고 한다.

```
PUBLIC SUB btnConnect_Click()
   DIM sName AS String
'만일 자료기지가 이미 열린 상태이면 그것을 닫도록 한다.
'TRY를 사용하면 어떠한 오유도 포착할수 있다.
'만일 닫아야 할 자료기지가 없으면 아무런 피해도 없이 코드는 그대로 수행된다.
   TRY $hConn.Close
   sName = txtName.Text
   WITH $hConn
      .Type = cmbType.Text              '리용하게 될 자료기지종류
```

```
        .Host = txtHost.Text                '주콤퓨터이름
        .Login = txtUser.Text               '사용자이름
        .Password = txtPassword.Text        '사용자암호
    END WITH
```

다음의 코드토막에서는 sName으로 지적한 자료기지가 존재하는가를 검사하고 존재하지 않으면 창조항목이 선택으로 된 경우에 창조한다.

```
    IF chkCreate.Value THEN
      $hConn.Open                        '접속열기
        '자료기지가 존재하는가를 보고 존재하지 않으면 그것을 봉사기에 추가
    IF NOT $hConn.Databases.Exist(sName) THEN
      $hConn.Databases.Add(sName)        '접속끌
    ENDIF
      $hConn.Close
    ENDIF
  '자료기지를 열 준비가 다 되였으므로 자료기지를 연다.
    $hConn.Name = sName
    $hConn.Open
  '두 홈의 enabled속성을 TRUE로 하고 Fmain홈의 밑에 있는 SQL질문창에는
  '표준적인 본문을 입력한다.
    frmDatabase.Enabled = TRUE
    frmRequest.Enabled = TRUE
    txtRequest.Text = "SHOW TABLES"
  '이 수속에서 TRY가 실패하거나 오유가 생기면 CATCH명령문은 오유통보문을
  현시한다.
  CATCH
      Message.Error(Error.Text)
  END
```

- *test단추를 찰칵하였을 때의 사건수속*

자료기지에 접속한 다음 사용자가 test단추를 찰칵하면 아래의 사건수속이 실행된다. 이 프로그람을 간단히 하기 위하여 일부를 주석을 리용하여 생략하였다. 필요하다면 주석을 해제하여 리용하시오.

```
  PUBLIC SUB btnCreate_Click()

    DIM hTable AS Table                 '국부적인 Table변수선언
```

```
                                    'Add메쏘드로 자료기지에 표를 추가
    'hTable = $hConn.Tables.Add("test")
    '이 코드는 우의 코드를 변화시킨것이다.
    hTable = $hConn.Tables.Add(txtName.text)
                              '우리가 만든 표에 마당을 첨가한다.
  '매 마당에 대하여 그의 이름과 자료형을 지적해준다.
  '또한 문자렬마당에 대하여서는 그 길이도 지적해준다.
    hTable.Fields.Add("id", gb.Integer)
    hTable.Fields.Add("firstname", gb.String, 16)
    hTable.Fields.Add("name", gb.String, 32)
    hTable.Fields.Add("birth", gb.Date)
    hTable.Fields.Add("active", gb.Boolean)
    hTable.Fields.Add("salary", gb.Float)
        '자료기지표의 기본색인을 지적한다. 이 경우에 id 마당은 유일적인
        '옹근수색인마당으로 된다.
    hTable.PrimaryKey = ["id"]
     'Update 메쏘드의 호출은 자료기지의 변화를 완료시켜준다.
    hTable.Update
   '지금까지 자료창조과정이 성과적으로 진행되였다는것을 통보.
    Message.Info("Table " & txtName.Text & " has been created.")
```

다음은 자료기지의 상태에 따라 단추들을 능동 또는 비능동상태로 만드는 코드를 작성한다.

실례로 자료기지표에 자료가 있으면 그 표에 다시 자료를 넣을 필요가 없으므로 fill단추는 비능동상태로 되고 대신 delete단추가 자료를 제거할수 있도록 능동상태로 된다.

만일 자료기지표가 제거되였다면 delete단추는 비능동상태로 되고 대신 Create단추가 능동상태로 되며 표는 창조되였는데 아무런 자료도 없으면 fill단추가 능동상태로 된다.

이것은 단추를 잘못 눌러 일어날수 있는 오유를 미리 방지한다.

```
    btnFill.Enabled = TRUE
    btnDelete.Enabled = TRUE
    btnCreate.Enabled = FALSE
  txtRequest.Text = "Show TABLES"          'SQL질문창에 표준값
  'CATCH명령문은 이 수속에서 어떠한 오유가 나타나면 오유통보문을 현시한다.
```

```
    CATCH
        Message.Error(Error.Text)
    END                                          'create단추끝
```

- *Delete단추를 찰칵했을 때의 사건수속*

자료기지접속이 이루어지면 사용자는 표를 삭제하기 위하여 Delete단추를 찰칵할수도 있다. 이때 다음의 사건수속이 호출된다.

여기서도 원래 프로그람을 쉽게 리해하도록 약간의 수정을 하였다.

```
    PUBLIC SUB btnDelete_Click()
      '$hConn.Tables.Remove("test")              '표의 제거
      $hConn.Tables.Remove(txtName.Text)
                                   '사용자에게 표가 제거되였다는 통보제시
      Message.Info("Table "& txtName.Text & " 제거되였음")
       '표가 삭제되였으므로 표를 다시 제거할수 없으며 자료를 채울수도 없다.
      '다시 표를 창조할수만 있다.
      btnDelete.Enabled = FALSE
      btnFill.Enabled = FALSE
      btnCreate.Enabled = TRUE
              '질문할 아무런 표도 존재하지 않으므로 질문칸을 빈것으로 한다.
      txtRequest.Text = ""
    CATCH
        Message.Error(Error.Text)
    END                                  'delete 단추끝
```

- *Fill단추를 찰칵하였을 때의 사건수속*

자료기지접속이 이루어지면 사용자는 표를 창조할수 있으며 거기에 일부 필요한 자료를 추가할수도 있다.

```
    PUBLIC SUB btnFill_Click()
          '첨수를 위하여 옹근수변수 iInd변수선언
          '결과를 보관하기 위한 결과객체변수선언
     DIM iInd AS Integer
     DIM rTest AS Result
      INC Application.Busy          '새치기를 방지하기 위하여 Busy기발설정
      $hConn.Begin                      '자료기지업무공정시작
        'rTest = $hConn.Create("test")       '먼저 자료기지표를 창조
```

```
    rTest = $hConn.Create(txtName.Text)
   FOR iInd = 1 TO 100              '다음 100개 레코드를 만들기 위한 순환조직
       rTest!id = iInd              '레코드의 첨수값 설정
                                 '5개 이름들중 란수적으로 선택된 이름이 들어간다
       rTest!firstname         =         ["Paul","Pie","Jaes","Ant","Mat"]
[Int(Rnd(5))]
       rTest!name = "Name #" & iInd        'last name에 첨수값이 련결된다
           '1970년 1월 1일에 1부터 1000까지의 수들중에서 우연수를 발생시켜
           '생일을 란수적으로 정한다.
       rTest!birth = CDate("01/01/1970") + Int(Rnd(10000))
                                 'active기발을 0과 1의 값중 아무것이나 설정한다
       rTest!active = Int(Rnd(2))
            '생활비를 1,000부터 10,000까지 값중에서 란수적으로 설정한다
       rTest!salary = Int(Rnd(1000, 10000))

       rTest.Update                    '갱신
   NEXT                                     '순환끝
  $hConn.Commit                             '자료기지에 대한 업무완료
                  '수속을 끝내기에 앞서 진행하여야 할 마지막 작업
  FINALLY
       DEC Application.Busy                 'busy 기발해제
                                            '사용자가 해야 할 일을 통보
       Message.Info(txtName.Text & " 에 자료가 첨부되였음")
                                            'SQL질문창에 SQL질문넣기
       txtRequest.Text = "select * from " & txtName.Text
       btnFill.Enabled = FALSE                  '단추동작상태 설정
       btnDelete.Enabled = TRUE
       btnCreate.Enabled = FALSE
       '오유가 발생하면 모든것을 취소하고 복귀하며 오유통보를 내보낸다.
  CATCH
       $hConn.Rollback
       Message.Error(Error.Text)
  END
```

- *Run단추를 찰칵하였을 때의 사건수속*

사용자는 홈의 밑에 있는 SQL질문창에 임의의 SQL질문을 입력하고 실행단추를 찰칵하여 질문의 결과를 볼수 있다.

```
PUBLIC SUB btnRun_Click()
  DIM rData AS Result           '질문결과를 얻기 위한 결과객체선언
  DIM hForm AS FRequest         'Frequest홈을 보여주기 위한 홈변수선언
 'Exec메쏘드를 리용하여 질문을 실행하고 그 결과를 결과객체변수 rData에 준다.
  rData = $hConn.Exec(txtRequest.Text)
           'Freques홈이 실체화될 때 자료기지조종자와 그 결과를 홈에 전달
  hForm = NEW FRequest($hConn, rData)
  hForm.Show                    '홈에 결과를 보여주기
  CATCH                         '오유가 발생하면 여기서 오유통보 현시
    Message.Error(Error.Text)
  END                                     'sql질문처리끝
```

- *Fmain홈이 열리거나 닫길 때의 사건수속*

프로그람이 실행되고 홈이 열릴 때마다 접속변수 $hConn를 실체화 할 필요가 있다.

```
PUBLIC SUB Form_Open()
  $hConn = NEW Connection
END

PUBLIC SUB Form_Close()
  $hConn.Close
END
```

2) Request홈에서의 코드작성

다음은 FRequest홈이 호출될 때 어떻게 결과객체자료를 보여주는가에 대하여 본다.

```
'Gambas class file
'변수소개
'하나는 접속을 위한 변수, 다른 변수는 결과자료를 위한 변수
PRIVATE $hConn AS Connection
PRIVATE $rData AS Result
```

- *Constructor사건수속*

이 사건수속은 Fmain홈이 Frequest홈을 호출할 때 접속조종자와 질문결과를 파라메터로 받기 위하여 사용된다.

```
PUBLIC SUB _new(hConn AS Connection, rData AS Result)
```

```
    'hConn 파라메터를 이 수속에서 $hConn 변수에 할당한다
  $hConn = hConn
                  'rData 파라메터를 $rData 변수에 할당
                  '변수할당이 끝나면 $가 앞머리에 붙은 변수들은
                  'constructor수속뿐아니라 전체클라스에서 리용할수 있다.
  $rData = rData
  RefreshTitle '홈창문에 제목을 현시하기위한 수속호출
                  'ReadData수속은 TableView 콘트롤을 배치하는네 리용
  ReadData
                       '탁상화면의 중심에 창문설치
 ME.Move(Int(Rnd(Desktop.W - ME.W)), Int(Rnd((Desktop.H - ME.H)))
END                               'constructor의 끝
```

- *창문제목을 표시하기 위한 수속*

```
PRIVATE SUB RefreshTitle()
DIM sTitle AS String    '국부변수소개
                     '접속파일이름을 본문과 련결하여 창문제목으로 한다.
  sTitle = ("SQL질문결과 ") & " - " & $hConn.Name
  ME.Title = sTitle
END
```

- *ReadData()수속*

여기서는 결과객체로부터 TableView콘트롤에 전달하게 될 자료의 구조를 정의한다.

즉 결과객체로부터 마당의 수를 결정하여 표의 렬수를 결정하며 적당한 렬이름과 자료형을 정의하여준다.

자료형은 WidthFromType()라는 다른 수속을 호출하여 결정한다.

```
PRIVATE SUB ReadData()
 '이 변수는 선언되였지만 사용하지 않았으므로 주석을 달아도 된다.
  DIM hTable AS Table
  DIM hField AS ResultField
  DIM sField AS String
  DIM iInd AS Integer
  DIM iLen AS Integer
                             '새치기를 피하기 위하여 buzy기발설정
  INC Application.Busy

  tbvData.Rows.Count = 0     'TableView콘트롤의 행수를 령으로 설정
```

```
                    '렬의 개수는 결과객체의 마당수와 같게 설정
tbvData.Columns.Count = $rData.Fields.Count
              '결과객체의 매 마당에 대한 이름과 자료형을 얻어내며
              'TableView의 렬마당과 자료형, 크기를 적당히 설정한다.
FOR EACH hField IN $rData.Fields
  WITH hField
                '이 주석을 단 부분은 실행추적을 위한 부분이다.
   'PRINT .Name; ": "; .Type; " "; .Length
                '마당이름으로 되는 렬이름을 설정
    tbvData.Columns[iInd].Text = .Name
 'FromType를 호출하여 마당이 어떤 자료형이며 그에 맞는 너비를 결정
   tbvData.Columns[iInd].Width=tbWidth(tbvData,.Type, .Length, .Na
  me)
  END WITH
  INC iInd              '첨수값증가
NEXT                    '결과객체에 대한 순환
                'TableView행의 수를 결과객체의 행수와 같게 설정
tbvData.Rows.Count = $rData.Count
FINALLY
   DEC Application.Busy      'busy기발해제
CATCH                   '오유처리
   Message.Error("질문을 실행할수 없음." & "\n\n" & Error.Text)
END                          'ReadData수속끝
```

- *TableView콘트롤의 Data사건수속*

TableView콘트롤의 Data사건은 자료가 현시될 때마다 동작한다.

이 사건에서는 결과객체의 자료 한행을 TableView콘트롤의 현재 행과 렬에 삽입한다.

data사건은 절대로 호출할수 없다.

TableView콘트롤을 리용하는 일반적인 방법에서 첫째는 묶음에 현시하려는 모든 자료를 적재한 다음 우의 ReadData에서 진행된것처럼 행과 렬로 된 TableView를 준비하는것이다.

```
PUBLIC SUB tbvData_Data(Row AS Integer, Column AS Integer)
$rData.MoveTo(Row)     '지적한 행으로 이동
                       '결과객체로부터 tableview의 칸에 자료를 입력
```

```
    tbvData.Data.Text = Str($rData[tbvData.Columns[Column].Text])
  END
```

- *홈의 크기를 변경시켰을 때의 사건수속*

```
  PUBLIC SUB Form_Resize()
    tbvData.Resize(ME.ClientW, ME.ClientH)
  END
```

- *마당의 자료형과 크기를 결정하는 함수*

```
  PRIVATE FUNCTION tbWidth(hCtrl AS control, iType AS Integer,
                          iLength AS Integer, sTitle AS String) AS
                          Integer
   DIM iWidth AS Integer
    SELECT CASE iType
      CASE gb.Boolean
        iWidth = hCtrl.Font.Width(Str(FALSE)) + 32
      CASE gb.Integer
        iWidth = hCtrl.Font.Width("1234567890") + 16
      CASE gb.Float
        iWidth = hCtrl.Font.Width(CStr(Pi) & "E+999") + 16
      CASE gb.Date
        iWidth = hCtrl.Font.Width(Str(Now)) + 16
      CASE gb.String
        IF iLength = 0 THEN iLength = 255
          iLength = Min(32, iLength)
          iWidth = hCtrl.Font.Width("X") * iLength + 16
        ENDIF
    END SELECT
    iWidth = Max(iWidth, hCtrl.Font.Width(sTitle) + 8)
    RETURN iWidth
  END
```

- *홈을 닫을 때의 사건수속*

```
  PUBLIC SUB Form_Close()
    ME.Close
  END
```

이 자료기지프로그람을 실행시키자면 먼저 PostgreSQL이나 MySQL과 같은 자료기지체계를 설치해야 한다.

다음 자료기지체계를 리용하기 위한 사용자식별표시와 암호를 설정한다. 그 다음 프로그람을 실행시키고 프로그람에서 자료를 창조하거나 질문을 실행시킬수 있다.

제 11 장. 마우스, 건반조종과 비트연산

이 장에서는 마우스와 건반을 조종하는 방법과 비트연산에 대하여 서술한다.

1. 마우스조종

마우스는 건반대신에 사용자가 입력할수 있는 가장 일반적인 입력장치이다.

창문에 의한 대면부가 출현한 때로부터 GUI는 많은 전진을 이룩하였지만 마우스의 진화에서는 별로 큰 전진이 없었다.

아마도 마우스에서 가장 큰 변화는 빛마우스와 무선마우스가 출현한것이라고 보아야 할것이다.

그러나 마우스의 기초적인 기능은 여전히 같다.

Gambas에서는 마우스에 대한 정보를 얻기 위하여 마우스클라스를 리용할수 있다.

이 클라스에는 다음과 같은 속성들이 있다.

Alt, Button, Control, Delta, Left, Meta, Middle, Normal, Orientation, Right, ScreenX, ScreenY, Shift, X, Y,

Alt, Control, Meta, Shift와 같은 속성들은 그 건반들이 눌리워있으면 TRUE를 돌려준다.

Normal은 임의의 다른 특수건반이 눌리워져있는가를 검사할수 있다.

Button으로 임의의 마우스단추를 찰칵하였는가를 검사할수 있다.

Left, Right, Middle속성으로 마우스의 해당 단추들이 눌리웠는가를 검사할수 있다.

이 모든 속성들은 단추가 눌리워있으면 TRUE를 돌려준다.

Orientation속성이 마우스굴개의 돌기방향(앞으로, 뒤로)을 결정한다면 Delta는 마우스굴개의 변위값을 돌려준다.

X와 Y가 창문에서의 상대적위치를 돌려준다면 ScreenX와 ScreenY는 탁상화면에서 마우스지적자의 절대적인 위치를 돌려준다.

마우스클라스에는 Move메쏘드만이 존재한다.

Move는 마우스지적자를 지적한 위치에로 이동시킨다.

조종할수 있는 마우스사건들은 Mousemove, MouseDown, Mouseup, Mousewheel이다.

이 사건들에서 조금이라도 변화가 있으면 그에 대응한 사건이 발생하며 그에 대하여 코드로 조종할수 있다.

마우스나 건반사건을 조종하는 코드를 작성하는데서 중요한것은 미리 정해진 상수를 항상 리용하는것이다.

마우스클라스는 마우스지적자의 형태를 지적하는 아래의 상수들을 리용할수 있다.

Arrow, Blank, Cross, Custom, Default Horizontal, Pointing, SizeAll, SizeE, SizeH, SizeN, SizeNE, SizeNESW, SizeNW, SizeNWSE, SizeS, SizeSE, SizeSW, SizeV, SizeW, SplitH, SplitV, Text, Vertical, Wait

[실습 1]

마우스조종법을 설명하기 위하여 간단한 프로그람을 작성하여보자.

IDE환경에서 MouseOps라는 새 프로젝트를 창조하고 그것을 도형사용자대면부방식, public된 콘트롤로, 번역가능하게 설정한다.

새 홈 Form1을 창조하고 시작홈으로 한다.

다음 Form1의 왼쪽웃구석에 《da》라는 이름을 가진 Drawingarea콘트롤을 배치하고 크기를 가로세로1inch크기로 설정한다.

다음은 코드작성을 진행한다.

먼저 프로그람이 실행되고 창문이 열리면 창문을 어떻게 지우고 끝내는가를 사용자에게 알려주는 제목띠의 표제를 설정한다.

다음 홈에 맞게 충분히 큰 그리기구역을 설정하고 고속완충속성을 TRUE로 하여 마우스로 그리는 과정을 사용자들이 볼수 있게 한다.

- *홈열기사건수속*

```
'Gambas class file
' 실행시 홈이 열리면 표제를 설정하고 그리기구역을 다시 설정한다.
PUBLIC SUB Form_Open()
  ME.Caption = "Press space to clear, mousewheel exits..."
  da.W = ME.W-5
  da.H = ME.H-5
  da.ForeColor = color.Black                '전경색을 검은색으로
          'cached 속성을 true로 설정하여 그림완충기를 리용할수 있게 한다.
  da.Cached = TRUE
  draw.Begin(da)                            '그리기 시작
             '그리기구역을 4분구로 나누기 위한 수평 및 수직선을 그린다.
  draw.Line(1,da.h/2,da.W-1,da.h/2)
  draw.Line(da.W/2, 1, da.W/2, da.H-1)
  draw.End
END
```

- *Mousemove사건수속*

마우스가 움직일 때 마우스가 그리기구역의 어느 분구에 있는가에 따라 마우스지적자의 모양이 달라지게 한다.

```
PUBLIC SUB da_MouseMove()
'왼쪽웃분구인가 검사
  IF mouse.X <= da.W/2 AND mouse.Y <= da.H/2 THEN
    da.Mouse = mouse.Pointing
   ELSE IF                              '왼쪽아래분구인가 검사
    mouse.X <= da.W/2 AND mouse.Y >= da.H/2 THEN
    da.Mouse = mouse.Cross
   ELSE IF                              '오른쪽웃분구인가 검사
    mouse.X >= da.W/2 AND mouse.Y <= da.H/2 THEN
    da.Mouse = mouse.SizeNWSE
   ELSE IF                          '나머지경우는 오른쪽아래분구인 경우
    mouse.X >= da.W/2 AND mouse.Y >= da.H/2 THEN
    da.Mouse = mouse.SizeNESW
   ELSE                         '이외의 경우 마우스형태를 표준형으로
    da.Mouse = mouse.Default
  ENDIF
```

다음 사용자가 어느 마우스단추를 찰칵하였는가를 검사하고 왼쪽단추를 찰칵한 경우에는 점을 그리고 오른쪽단추를 찰칵한 경우에는 작은 통을 그리게 한다.

```
  IF Mouse.Left THEN
    Draw.Begin(da)                           '그리기시작
    Draw.Point(Mouse.X, Mouse.Y)             '작은점 그린다.
    Draw.End
   ELSE                             '오른쪽단추를 찰칵한 경우
    Draw.Begin(da)
    Draw.Rect(Mouse.X, Mouse.Y, 3, 3)
    Draw.End
  ENDIF
END
```

- *Mousewheel 사건수속*

사용자가 마우스굴개를 움직이면 프로그람을 끝내게 한다.

```
PUBLIC SUB da_MouseWheel()
  ME.Close
END
```

마지막으로 사용자가 임의의 건반을 누르면 그리기구역을 지우고 홈을 열 때와 꼭 같은 수속을 호출하여 분구를 다시 그리도록 한다.

```
PUBLIC SUB Form_KeyPress()
   da.Clear
   Form_Open()
END
```

실행결과

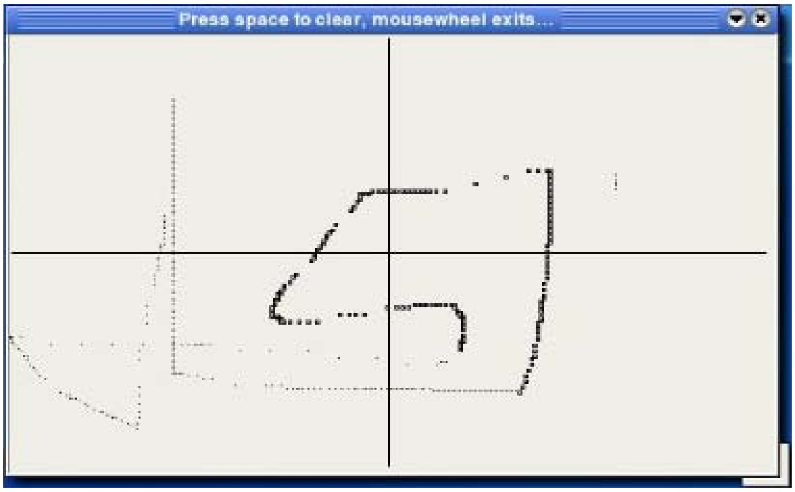


그림 11-1. MouseOps프로그람의 실행결과

2. 건반조종

건반조종사건들에 대한 정보를 얻기 위하여 건반클라스를 리용할수 있다..

건반클라스의 속성들로서는 Alt, Code, Control, Meta, Normal, Shift, State, Text 등이 있다.

State는 특수건반들이 눌리웠는가를 검사할 때 리용할수 있으며 Text는 주어진 건반에 해당한 글자를 돌려준다.

이밖의 속성들은 앞에서 취급한 마우스클라스에서 리용한 속성과 꼭같다.

건반클라스는 다음과 같은 미리 정해진 상수들을 가지고있다.

위치지적자건반으로서 Up, Down, Left, Right가 있다.

기능건으로서 F1, F2, F3, F4, F5, F6, F7, F8, F9, F10, F11, F12이 있다.

shift를 누른 기능건으로서 F13, F14, F15, F16, F17, F18, F19, F20, F21, F22, F23, F24이 있다.

이외의 다른 건반들로서 다음과 같은 건반들이 있다.

BackSpace,	BackTab,	CapsLock,	Delete,	End
Enter,	Esc,	Escape,	Help,	Home
Insert,	Menu,	NumLock,	PageDown,	PageUp
Pause,	Print,	Return,	ScrollLock,	Space
SysReq	Tab			

이미 앞에서 지적하였지만 절대로 옹근수건반값들을 직접 리용하지 말고 Gambas의 상수들을 사용해야 한다.

옹근수변수를 선언하고 다음과 같은 형식으로 건반상수값을 얻을수 있다.

```
DIM MyInteger AS Integer
MyInteger = Key [ Key AS String ]
```

[실습 2]

건반조종방법을 련습하기 위하여 KbdOps라는 이름을 가진 프로그람을 간단히 작성하여보자.

이 프로그람에서는 두개의 표식콘트롤 TextLabel과 Label만을 사용한다.

먼저 TextLabel을 TextLabel1로, 그의 높이를 ½inch정도로, 너비를 홈의 너비만큼으로 설정하여 배치한다.

TextLabel1의 Alignment속성을 center로 설정한다.

Label의 이름을 Label1로 한 다음 TextLabel1의 아래에 배치한다.

크기는 너비가 2inch로, 높이가 ½inch로 설정한다.

또한 Label의 Text속성은《Press any key…》로, 콘트롤의 서체크기는 14로, 글체는 굵은체로 설정한다.

다음의 코드를 삽입한다.

```
' Gambas class file
'어떤 건반이 눌렀는가를 검사한다
PUBLIC SUB Form_KeyRelease()
  SELECT key.code                    '건반을 누를 때마다 코드를 검사
    CASE key.Tab                           '만일 tab 건이라면
      textlabel1.Text = "Key code is: " & key.Code & ",TAB."
    CASE key.BackSpace                 'backspace건이라면
      textlabel1.Text = "Key code is: " & key.Code & ",Backspace."
```

```
        CASE key.CapsLock                 'caps lock건이라면
          textlabel1.Text = "Key code is: " & key.Code & ",CapsLock."
        CASE key.F1                           '기능건인 경우
          textlabel1.Text = "Key code is: " & key.Code & ", F1 key."
        CASE key.F2
          textlabel1.Text = "Key code is: " & key.Code & ", F2 key."
        CASE key.F3
          textlabel1.Text = "Key code is: " & key.Code & ", F3 key."
        CASE key.F4
          textlabel1.Text = "Key code is: " & key.Code & ", F4 key."
        CASE key.F5
          textlabel1.Text = "Key code is: " & key.Code & ", F5 key."
        CASE key.F6
          textlabel1.Text = "Key code is: " & key.Code & ", F6 key."
        CASE key.F7
          textlabel1.Text = "Key code is: " & key.Code & ", F7 key."
        CASE key.F8
          textlabel1.Text = "Key code is: " & key.Code & ", F8 key."
    '우와 다른 경우 표시할수 있는건인가 검사하고
    '표시할수 있으면 그 글자와 코드값을 보여준다.
        CASE ELSE
          IF key.Code > 32 AND key.Code < 128 THEN
            textlabel1.Text="keycode is:" & key.Code & "," &
Chr(key.Code)
            ELSE          '표시할수 없는 건인 경우 코드값을 돌려준다.
              textlabel1.Text = "key code is: " & key.Code
          ENDIF
      END SELECT
    END
                'shift, control, alt와 같은 건들이 눌리웠는가를 검사
    PUBLIC SUB Form_KeyPress()
      IF key.Shift THEN
          textlabel1.Text = "Key code is: " & key.Code & ", SHIFT"
        ELSE IF key.Alt THEN
          textlabel1.Text = "Key code is: " & key.Code & ", ALT"
```

```
        ELSE IF key.Control THEN
          textlabel1.Text = "Key code is: " & key.Code & ", CTRL"
        ELSE IF key.Code > 32 AND key.Code < 128 THEN
          textlabel1.Text  =  "key  code  is:  "  &  key.Code  &  ","  &
Chr(key.Code)
        ELSE IF key.Esc THEN                        'ESC건이 눌리웠으면
          textlabel1.Text = "key code is: " & key.Code & ", ESC"
        ELSE
            textlabel1.Text = "key code is: " & key.Code
      ENDIF
   END

   PUBLIC SUB Form_Open()
      ME.Caption = " Keyboard Operations "
   END
```

실행결과



그림 11-2. KbdOps프로그람

3. 비트연산

비트(bit)는 콤퓨터연산에서 리용되는 가장 작은 단위이다.

비트들은 byte라고 부르는 기억단위를 이룬다.

8bit체계에서 한byte는 8bit로 구성된다.

오늘 사용되는 현대적인 콤퓨터체계에서 조작체계는 32bit 혹은 64bit를 단어크기로 한다.

단어크기는 주소기억공간이 클수록 증가된다.

이와 같이 기억기관리, 마스크조종, 수학적계산 등에는 비트연산이 필요하게 된다.

비트연산을 위하여 8개의 함수들이 준비되여있다.

비트연산함수들은 모두 두개의 파라메터를 가지는데 첫 파라메터는 0~255사이의 수(1byte가 표시할수 있는 수)를 의미하며 두번째파라메터는 비트자리번호 혹은 개수를 의미한다. 비트자리번호는 아래의 그림과 같이 정한다.

7	6	5	4	3	2	1	0

- BTst

> 형식
>
> BTst(수, n)

수의 n번째비트가 1이면 TRUE, 0이면 FALSE를 돌려준다.

- BClr

> 형식
>
> BClr(수, n)

수의 n번째 비트값을 0으로 설정하여 돌려준다.

- Bset

> 형식
>
> BSet(수, n)

수의 n번째 비트값을 1로 설정하여 돌려준다.

- BChg

> 형식
>
> BChg(수, n)

수의 n번째 비트값을 반대로 설정하여 돌려준다.

- Shl

> 형식
>
> Shl(수, n)

수를 n비트수만큼 왼쪽으로 밀기한 값을 돌려준다.

- Shr

> 형식
>
> Shr(수, n)

수를 n비트수만큼 오른쪽으로 밀기한 값을 돌려준다.

- Rol

형식 Rol(수, n)

수를 n비트수만큼 왼쪽으로 회전시킨 값을 돌려준다.

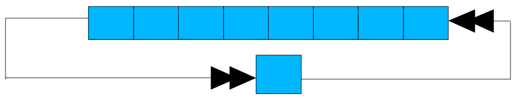
- Ror

형식 Ror(수, n)

수를 n비트수만큼 오른쪽으로 회전시킨 값을 돌려준다.

Rol과 Ror함수들은 다음과 같이 작용한다.

Rol연산



Ror연산

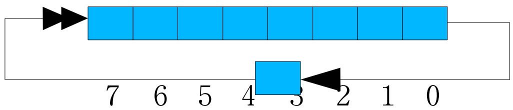


그림 11-3. Rol과 Ror연산순서

Rol은 먼저 한비트를 왼쪽으로 한자리씩 밀기한다.

이때 0번자리에는 0값이 아니라 7번 비트값이 들어온다.

Ror는 Rol과 꼭같이 동작하지만 방향이 반대이다.

즉 한비트를 한자리씩 오른쪽으로 밀며 0번 비트값이 7번 비트로 간다.

[실습 3]

비트함수들이 어떻게 동작하는가를 보기 위하여 간단한 프로그람을 작성하여보자.

Gambas IDE에서 BitOps라는 새 프로젝트를 창조한다.

이 프로젝트를 도형대면부방식으로, 번역할수 있는 상태로, 콘트롤들을 public로 설정한다.

Form1을 시작홈으로 한다.

이 프로그람은 첫 본문칸에 0~255사이의 임의의 옹근수를 입력하면 그 수의 16진수와 2진수표시를 보여준다.

또한 아래의 본문칸에는 2진수에 대한 비트번호(또는 비트개수)를 입력한다. 그 다음 아래의 단추를 찰칵하는데 따라 임의의 연산을 진행한다.

연산결과는 2진수값으로 표현되며 이전에 입력한 수값과 비교하여 볼수 있다.

이 프로그람을 위하여 TextLabel1부터 Textlabel7까지의 7개의TextLabel과 TextBox1과 TextBox2라는 본문칸, RolBtn, RorBtn, ShlBtn, ShrBtn, ClearBtn, TestBtn, SetBtn,ChangeBtn, QuitBtn라는 9개의 단추를 아래의 그림과 같이 배치한다.



그림 11-4. 설계시의 BitOps프로그람

홈을 열 때의 사건수속

```
' Gambas class file
PUBLIC SUB Form_Open()
   ME.Caption = " bitOpns "
END
```

Quit단추를 찰칵할 때의 사건수속

```
PUBLIC SUB QuitBtn_Click()
   ME.Close
END
```

Textbox2값이 변화될 때의 사건수속

TextBox2마당에 어떤 값을 넣어 비트를 변화시키면 변화된 값으로 홈을 수정할 필요가 있다. 이것은 다음의 사건수속으로 처리할수 있다.

```
PUBLIC SUB TextBox2_Change()
   form1.Refresh                '비트값이 변화될 때마다 홈을 다시 현시
END
```

Textbox1의 초점을 변화시킬 때의 사건수속

사용자가 TextBox1에 옹근수값을 지적하고 TAB건을 눌러야 LostFocus사건이 발생하며 입구값에 반응한다.

```
PUBLIC SUB TextBox1_LostFocus()
   DIM myInt AS Integer
   myInt = CInt(textbox1.text)      '기호렬값을 옹근수로 변환
  IF Int(myInt >= 0) AND (myInt < 256) THEN '값이 8bit범위의 값인가 검사
      Label2.Text = "Hex: " & Hex(myInt)      '16진수값으로 변환
      Label3.Text = "Bin: " & Bin(myInt)      '2진값으로 변환
      Label7.Text = Str$(myInt) '
      form1.Refresh                     '홈을 다시 표시한다.
```

```
    ENDIF
  END
```

Clear단추를 찰칵할 때의 사건수속

clear단추를 찰칵하면 옹근수에서 해당 비트값을 지운다.

```
  PUBLIC SUB ClearBtn_Click()
    DIM i AS Integer
    DIM result AS Integer
    DIM myInt AS Integer
    myInt = CInt(textbox1.text)                '기호렬값을 옹근수로 변환
                  '옹근수값이 0-255사이이면 처리를 계속하고 그렇지않으면 무시
    IF Int(myInt >= 0) AND (myInt < 256) THEN
      result = CInt(TextBox2.Text)             '기호렬값을 옹근수로 변환
      i = BClr(myInt, result)                     '비트값 지우기
      label5.Text = "Bclr: " & Bin(i)             '표식콘트롤값 수정
      label7.Text = Str$(i)
    ENDIF
  END
```

Set단추를 찰칵할 때의 사건수속

Set단추를 찰칵하면 옹근수에 대하여 지적된 비트를 1로 설정한다.

이것은 우에서 본 clear단추의 경우와 반대이다.

```
  PUBLIC SUB SetBtn_Click()
    DIM i AS Integer
    DIM result AS Integer
    DIM myInt AS Integer

    myInt = CInt(textbox1.text)                '기호렬값을 옹근수로 변환
             '옹근수값이 0~255이면 다음단계를 진행하고, 그렇지 않으면 무시
    IF Int(myInt >= 0) AND (myInt < 256) THEN
      result = CInt(TextBox2.Text)       '기호렬값을 옹근수로 변환
      i = BSet(myInt, result)            '비트설정
      label5.Text = "Bset: " & Bin(i)         'label들을 수정
      label7.Text = Str$(i)
    ENDIF
  END
```

Change단추를 찰칵했을 때의 사건수속

사용자가 Change단추를 찰칵하면 Bchg함수의 거꿀연산이 수행된다.

```
PUBLIC SUB ChangeBtn_Click()
  DIM i AS Integer
  DIM result AS Integer
  DIM myInt AS Integer
  myInt = CInt(textbox1.text)
  IF Int(myInt >= 0) AND (myInt < 256) THEN
    result = CInt(TextBox2.Text)
    i = BChg(myInt, result)
    label5.Text = "Bchg: " & Bin(i)
    label7.Text = Str$(i)
  ENDIF
END
```

Shl단추를 찰칵하였을 때의 사건수속

```
PUBLIC SUB SHLBtn_Click()
  DIM i AS Integer
  DIM result AS Integer
  DIM myInt AS Integer
  myInt = CInt(textbox1.text)
  IF Int(myInt >= 0) AND (myInt < 256) THEN
    result = CInt(TextBox2.Text)
    i = Shl(myInt, result)
    label5.Text = "SHL: " & Bin(i)
    label7.Text = Str$(i)
  ENDIF
END
```

Shr단추를 찰칵하였을 때의 사건수속

```
PUBLIC SUB SHRBtn_Click()
  DIM i AS Integer
  DIM result AS Integer
  DIM myInt AS Integer
  myInt = CInt(textbox1.text)
  IF Int(myInt >= 0) AND (myInt < 256) THEN
```

```
      result = CInt(TextBox2.Text)
      i = Shr(myInt, result)
      label5.Text = "SHR: " & Bin(i)
      label7.Text = Str$(i)
    ENDIF
  END
```

Rol단추를 찰칵하였을 때의 사건수속

```
  PUBLIC SUB ROLBtn_Click()                    '왼쪽회전
    DIM i AS Integer
    DIM result AS Integer
    DIM myInt AS Integer
    myInt = CInt(textbox1.text)
    IF Int(myInt >= 0) AND (myInt < 256) THEN
      result = CInt(TextBox2.Text)
      i = Rol(myInt, result)
      label5.Text = "ROL: " & Bin(i)
      label7.Text = Str$(i)
    ENDIF
  END
```

Ror단추를 찰칵하였을 때의 사건수속

```
  PUBLIC SUB RORBtn_Click()                      '오른쪽회전
    DIM i AS Integer
    DIM result AS Integer
    DIM myInt AS Integer
    myInt = CInt(textbox1.text)
    IF Int(myInt >= 0) AND (myInt < 256) THEN
      result = CInt(TextBox2.Text)
      i = Ror(myInt, result)
      label5.Text = "ROR: " & Bin(i)
      label7.Text = Str$(i)
    ENDIF
  END
```

Test단추를 찰칵하였을 때의 사건수속

```
  PUBLIC SUB TestBtn_Click()
    DIM i AS Boolean
```

```
    DIM result AS Integer
    DIM myInt AS Integer
    myInt = CInt(textbox1.text)
    IF Int(myInt >= 0) AND (myInt < 256) THEN
       result = CInt(TextBox2.Text)
       i = BTst(myInt, result)
       label5.Text = "Btst: " & i
    ENDIF
END
```

실행결과



그림 11-5. bitOps프로그람의 실행결과

제 12 장. 프로그람의 국제화

Gambas는 최초부터 국제적인 통신과 교류를 보장할수 있도록 설계되였다.

따라서 Gambas는 개발자들에게 모든 나라의 리용자들을 만족시키는 대면부를 가진 프로그람을 작성할수 있는 가능성을 주고있다.

그리하여 필요한 언어로 쉽게 변환되는 프로그람들이 개발되고있다.

이 장에서는 모든 Gambas프로그람개발자들이 국제화가 실현된 프로그람을 개발하는데서 알아야 할 기초적인 지식을 주게 된다.

1. 프로그람의 국제화

프로그람의 국제화는 그 어떤 기술적변화를 주지 않아도 여러 언어와 지역에 적용될수 있는 응용프로그람을 설계하는 공정을 말한다.

때때로 국제화라는 단어를 i18n으로 간략하여 표현하기도 한다. 왜냐하면 국제화(Internationalization)라는 단어의 시작과 끝사이에 18개의 글자가 있기때문이다.

지역적자료가 첨부된 국제화된 프로그람은 세계의 어떤 지역에서도 실행될수 있다.

상태통보와 도형대면부요소들에서 나오는 본문들은 프로그람에서 완전히 고정된 코드가 아니라 원천코드밖에 보관되여 해당 언어 및 지역에 따라 동적으로 읽어진다.

즉 새 언어를 선택한다고 하여 프로그람을 다시 콤파일할것을 요구하지 않는다.

결국 날자라든가 화폐와 같은 자료들은 말단사용자의 지역과 언어선택에 따라 자동적으로 나타난다.

프로그람의 국제화를 실현하자면 세계 모든 지역에서 사용하는 글자들을 표현할수 있어야 한다. 이를 위하여 국제적으로 UCS(만능문자모임)와 UNIcode(유니코드)를 제정하였다. ISO/IEC10646는 만능문자모임(UCS)이라고 하는 대단히 큰 문자모임을 제정하였는바 이것은 세계의 글쓰기체계의 대부분을 포함하고있다.

UCS의 매 글자들에는 유니코드가 대응되게 된다.

유니코드는 현재 약 100 000개의 글자들을 정의하고있지만 1 114 112개의 코드구역을 가지고있다. 코드구역은 0부터 16까지의 번호가 붙은 17개의 면들로 되여있으며 매 면은 216개의 글자코드로 이루어져있다.

0면에는 일반적으로 유니코드가 나오기 이전에 개발자들이 사용하던 기초적인 문자들을 포함하고있으며 다른 면들에서 문자들은 《서쪽에서 동쪽으로》의 순서로 분포되여있다.

따라서 0~127값은 이미 잘 알려진 ASCII코드값을 가지며 128~255까지의 값은 ISO-Latin-1 문자에 해당한 값을 가지게 된다. 그 다음 문자들은 유럽을 가로질러 동

쪽으로(그리스문자, 카릴로스문자) 이동하여 중동지역(아랍어, 헤브라이어 등)을 거쳐 인디아와 동남아시아를 거쳐 중국, 일본과 조선문자들로 끝난다.

오늘까지 UCS와 유니코드에 대한 수정 및 보충사업은 조화롭게 진행되고있으며 따라서 문자수집과 코드표현은 동시에 진행되고있다.

2. 프로그람의 국제화실현방법

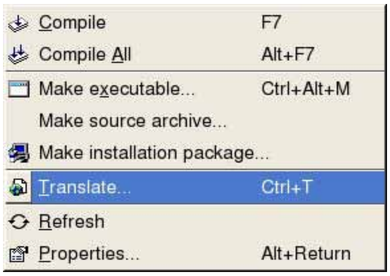


그림 12-1. Translate항목선택

프로그람의 국제화를 실현하려면 IDE에서 Gambas 프로젝트를 열고 그림과 같이 Project/Translate...를 찰칵한다.

다음 Translate대화창의 첫 복합칸에서 번역하려는 언어를 선택한다. 그러면 프로젝트의 대면부콘트롤표제들과 통보문들이 모두 목록으로 나타난다.

여기서 번역하려는 문자렬을 선택하면 untranslated strings목록에 선택한 문자렬이 나타난다.



그림 12-2. Translate대화칸

대화창의 밑에 있는 본문마당에 해당한 번역내용을 편집하여 넣는다. 개발하는 응용프로그람의 모든 문자렬을 번역할 때까지 이 공정을 반복한다. 이것이 끝나면 Close단추를 찰칵한다. 이러한 문자렬번역공정은 여러번에 걸쳐 진행될수 있으므로 편집 및 관리를 위한 여러가지 기능들이 필요하다.

따라서 Translate대화칸의 꼭대기에는 다음과 같은 기능을 수행하는 아이콘들이 차례로 있다.

- 현재 번역한것을 보관
- 번역한것을 재적재(주의: 재적재할 때 번역하던 내용은 잃게 된다.)
- 현재 번역한것을 삭제
- 번역한 내용의 부본창조
- 현재 번역한것을 외부파일로 출력
- 번역파일들을 현재 번역한 내용과 결합
- 매 문자들이 보존되였는가를 검사

번역과정에 일부 문자렬은 번역하지 말아야 하는 경우도 있다.

이때는 번역되는 문자렬에 덜기기호를 넣어 번역되지 않는 문자렬을 지적할수 있다.

번역결과는 해당 프로젝트의 .lang등록부에 *.po파일에 보관된다.

*.po파일의 이름은 번역된 언어에 관계된다.

실례로 프랑스어로 번역된 파일은 fr.po라는 이름으로 된다.

앞으로 Gambas2가 출하될 때에는 다음과 같은 문제들이 완전히 해결될것으로 본다.

모든 통보들과 아이콘들, 사람들이 읽을수 있는 내용들을 외부자원파일들에 보관하며 쉽게 번역할수 있다.

번역된 통보들이나 자원파일들은 현재 언어와 지역적인 설정에 따라 동적으로 응용프로그람에 적재되게 된다. 날자/시간, 분류차례, 수자와 화폐형식 등은 목적하는 언어에 따른다. 분류차례는 사용자의 언어에 따른다.

사용자는 자기가 목적하는 언어로 정확히 입력할수 있으며 현시할수 있게 된다.

또한 문자들은 목적하는 플레트홈의 다국어파일체계로부터 읽어들일수 있으며 쓰기도 할수 있다.

개발된 응용프로그람의 국제화수준을 평가하는 기준은 다음과 같다.

- 프로그람이 국제화를 고려하였는가?
- 사용자가 건반으로 지름건을 조합할수 있는가?
- 사용자가 국제화형식을 사용하여 날자, 시간, 화폐, 수자들을 입력할수 있는가?
- 사용자가 글자들로 된 본문을 자르기/복사할수 있는가?
- 응용프로그람이 지역화조종체계에서 정확히 동작하는가?
- 응용프로그람이 서로 다른 형태의 장치들에서 정확하게 동작할수 있는가?
- 사용자가 유럽의 력점있는(혹은 아시아의 두byte문자)문자들을 문서와 대화칸에서 입력할수 있는가?
- 사용자가 력점있는 문자（혹은 두byte문자）들로 된 파일들을 보관하고 인쇄할수 있는가?
- 목적하는 언어로 창조된 문서들이 반대로 다른 언어들에서 열릴수 있는가?

제 13 장. Gambas 와 VB 의 차이점

Gambas와 VB는 서로 비슷하며 언어상 1대 1대응관계에 있는것도 많지만 차이점도 적지 않다.

1. 차이점

VB에서는 홈을 정의하는 파일안에 홈객체와 클라스코드가 함께 섞여있다.
그러나 Gambas에서는 홈과 클라스를 각각 .form, .class로 서로 구분한다.
파일확장자들도 차이난다.

VB	Gambas	파일형태
.vbp	.project(하나의 등록부에 하나의 프로젝트 대응)	프로젝트파일
.bas	.module	모듈
.cls	.class	클라스파일
.frm	.form	홈파일
.frx		2진원천파일

Gambas프로젝트는 프로젝트파일과 다른 모든 파일들을 포함한 하나의 등록부로 정의된다. 그러나 VB에서는 하나의 등록부에 여러개의 프로젝트들을 포함할수 있다.

VB에서는 화면자리표의 단위가 《twip》(1twip=1/1440inch)이지만 Gambas에서는 《pixel》로 되여있다.

Gambas에서 str$(), val(), cstr$()와 같은 변환함수들은 VB에서는 전혀 볼수 없는 지역적특성을 가진다.

Gambas에서는 integer, string과 같은 단순자료형파라메터들이 함수나 수속에 값으로 전달된다. 그러나 VB에서는 모든 파라메터들이 byval을 지적하지 않는 한 참고형태로 전달된다. 한편 객체자료형파라메터는 두 언어에서 다 참고형태로 전달된다.

그러므로 VB프로그람을 Gambas에 이식하려 할 때에는 이런 점을 주의하여야 한다.

Gambas에서 프로젝트범위의 대역변수의 소개는 VB와 완전히 차이난다.

먼저 대역적인 클라스를 만들고 그 클라스안에 static, public형태로 대역변수를 소개한다. 다음 프로젝트의 임의의 장소에서는《Global.변수이름》형태로 변수를 리용할수 있다.

VB모듈에서는 option explicit를 선언하지 않으면 변수소개를 하지 않아도 된다.

그러나 Gambas에서는 항상 변수소개를 할것을 요구한다.

Gambas에는 VB홈콘트롤속성인 index속성에 대응한 속성이 없다.

그러므로 콘트롤묶음을 창조하자면 코드를 리용하여야 한다.

또한 홈에서 콘트롤을 복사하여 붙이기하면 VB에서처럼 콘트롤묶음이 생기지 않고 원래 콘드롤과 비슷한 이름을 가진 새로운 콘트롤이 생긴다.

Gambas에서는 label콘트롤에 투명한 배경을 줄수 없다.

Gambas에서 mousemove사건은 마우스단추를 누른 상태에서만 일어난다. 그러나 DrawingArea콘트롤에서는 례외적이다.

DrawingArea콘트롤에서는 Tracking속성에 따라 마우스단추를 누르지 않아도 mousemove사건을 일으킬수 있다.

VB에서는 두 문자렬을 련결시키기 위하여 《+》연산자를 리용할수 있다. 그러나 Gambas에서는 《+》연산자가 수학연산에만 리용되므로 두 문자렬을 련결하자면 《&》연산자를 리용해야 한다.

Gambas에서는 print명령문 음에《:》기호를 명령문분리기호로 사용하지 말아야 한다. 즉 코드작성시 print명령문이 있는 행에는 다른 명령문을 함께 쓰지 말아야 한다.

VB에서는 mid$()가 부분기호렬을 교체하는 함수로 리용되지만 Gambas에서는 부분기호렬을 따내는 함수로 리용된다.

실례로

VB에서는

```
mystring=" Is this Gambas?"
Mid$(mystring,4,4)=" that"
Print mystring
```

결과

```
Is that Gambas?
```

Gambas에서 같은 결과를 얻자면

```
mystring=" Is this Gambas?"
mystring=left$(mystring,3) & "that" & mid$( mystring,8)
Print mystring
```

Gambas에서는 오유조종에 Goto명령문을 쓸수 없다.

오유조종을 위해서는 Catch, Finally, Try명령문을 리용해야 한다.

Gambas에는 VB에서 볼수 없는 콘트롤그룹이라는 개념이 있다.

이것을 리용하면 같은 사건수속을 가지는 서로 다른 여러개의 콘트롤들을 그룹으로 묶어 코드작성을 간소화할수 있다.

Gambas에서는 New명령문을 리용하여 콘트롤과 안내문들을 동적으로 창조할수 있다.

그러나 VB에서는 콘트롤들을 동적으로 참조할수 없다.

Gambas에서는 콘트롤들에 대한 마우스효과를 위하여 Enter, leave사건을 리용한다.

Gambas에서 2진파일을 읽어 처리하자면 open명령문에서 big와 little을 지적하여 주어야 한다.

Gambas는 프로젝트를 충분히 국제화할수 있다.

Gambas는 그 개발환경도 자체언어로 작성된 개방형프로그람이므로 개발자의 요구에 맞게 개발환경을 변화시킬수 있다.

2. 기능은 같지만 언어적표현이 다른것

같은 기능에 대하여 서로 다른 방식으로 처리하는것을 간단히 표로 보여준다.

VB	Gambas
Sub/End Function	End
End Sub/Exit Function	Return 함수값을 귀환하는 방법도 차이난다.
End (프로그람끝내기)	Quit
On Error	Try,Catch,Finally
Messagebox	Message, Message.info
Inputbox함수	Inputbox클라스
Doevents	Wait
기호렬에〃를 삽입하자면 〃〃	기호렬에〃를 삽입하자면 \〃
Vscrollbar 와 Hscrollbar콘트롤	Scrollbar콘트롤
건반과 마우스사건수속 파라메터들 실례 form_mouseup(button as integer, shift as integer, x as single, y as single)	Key와 mouse클라스를 리용 실례 mouse.button,mouse.shift mouse.x, mouse.y, key.code

Gambas프로그람작성기초

집 필	조영실	심 사	홍성임, 로순영
편 집	강혜경	교 정	여은정
장 정	서경애	콤퓨터편성	여은정
낸 곳	교육성 교육정보쎈터	인쇄소	교육성 교육정보쎈터
인 쇄	주체 97(2008)년 8 월 10 일	발 행	주체 97(2008)년 8 월 20 일

교-08-1586